T.Y.B.Sc. COMPUTER SCIENCE

SEMESTER - V

Subject Code: USCS502

Subject Name: Information Network Security

(NEW SYLLABUS W.E.F. 2023-2024)

Course Writer:

Prof. Hasan Phudinawala

RFC 2828 defines **information** as "facts and ideas, which can be represented (encoded) as various forms of data," and **data** as "information in a specific physical representation, usually a sequence of symbols that have meaning; especially a representation of information that can be processed or produced by a computer."

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).

These three concepts form what is often referred to as the **CIA triad**

1. **Confidentiality**: Confidentiality: Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.

This term covers two related concepts:

Data1 confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals.

Privacy: Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

2. Integrity: Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information. This term covers two related concepts:

Data integrity: Assures that information and programs are changed only in a specified and authorized manner.

System integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

3. **Availability**: Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system. Assures that systems work promptly and service is not denied to authorized users.

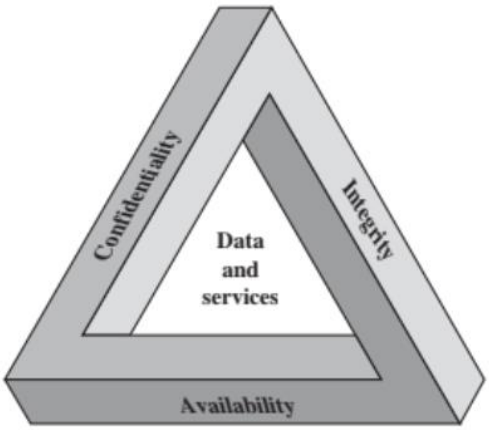


Figure 1.1 The Security Requirements Triad

Although the use of the CIA triad to define security objectives is well established, some in the security field feel that additional concepts are needed to present a complete picture. Two of the most commonly mentioned are as follows:

• **Authenticity**: The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator. This means verifying that users are who they say they are and that each input arriving at the system came from a trusted source.

• **Accountability**: The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity. This supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action. Because truly secure systems are not yet an achievable goal, we must be able to trace a security breach to a responsible party. Systems must keep records of their activities to permit later forensic analysis to trace security breaches or to aid in transaction disputes.

The Osi Security Architecture

ITU-T3 Recommendation X.800, Security Architecture for OSI, defines such a systematic approach.4 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. For our purposes, 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.

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 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.

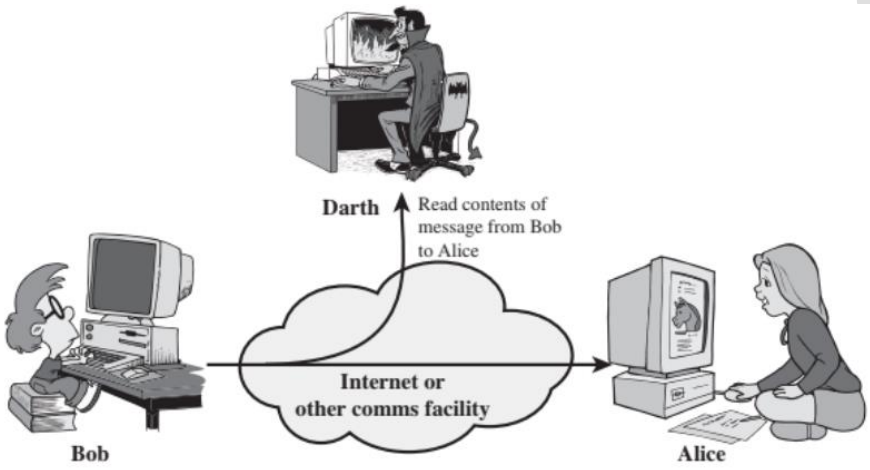
Security service: A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

Security Attacks

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 the release of message contents and traffic analysis.

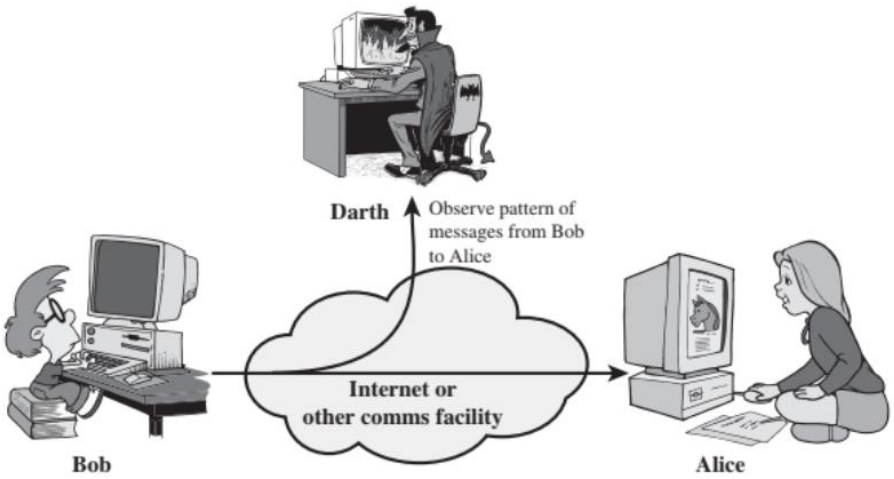
1. Release of message contents



(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.

2. Traffic Analysis



(b) Traffic analysis

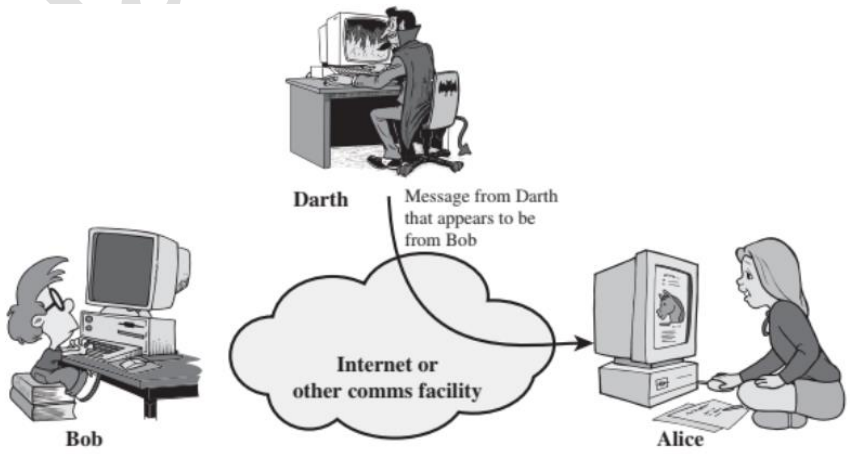
Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent 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.

Active Attacks

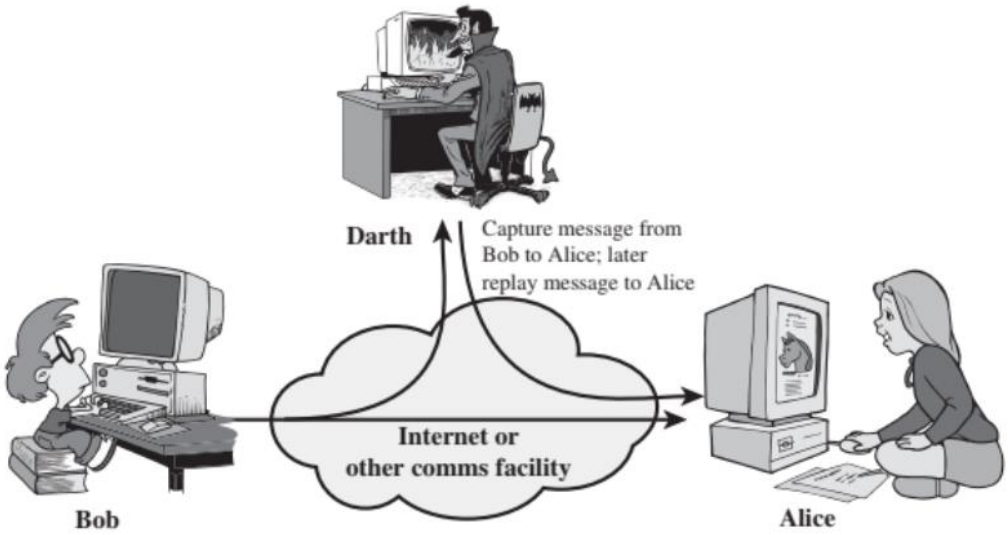
Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: **masquerade, replay, modification of messages, and denial of service.**

1. **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.

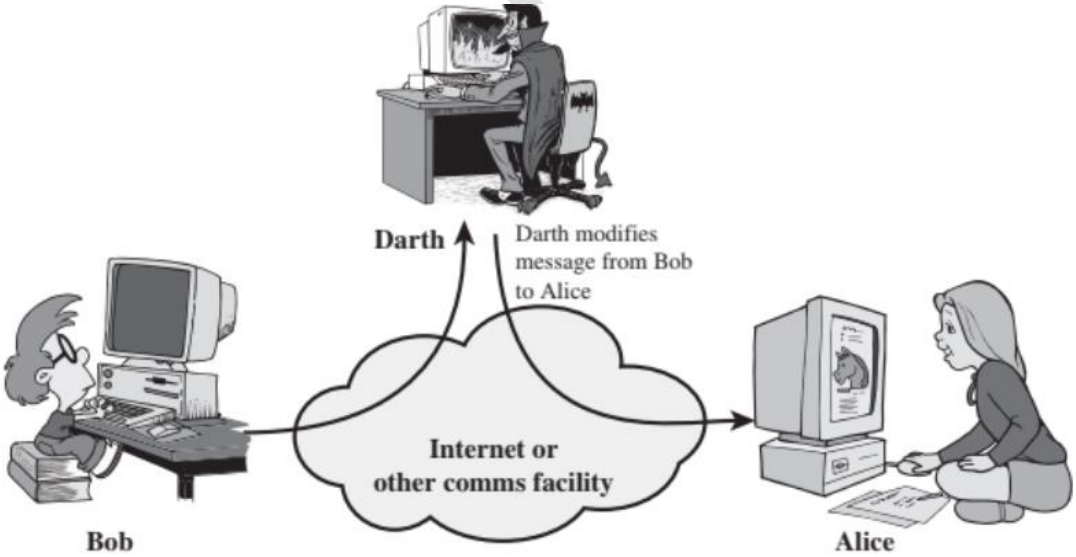


(a) Masquerade

2. Replay involves the passive capture of a data unit and its subsequent retrains mission to produce an unauthorized effect.



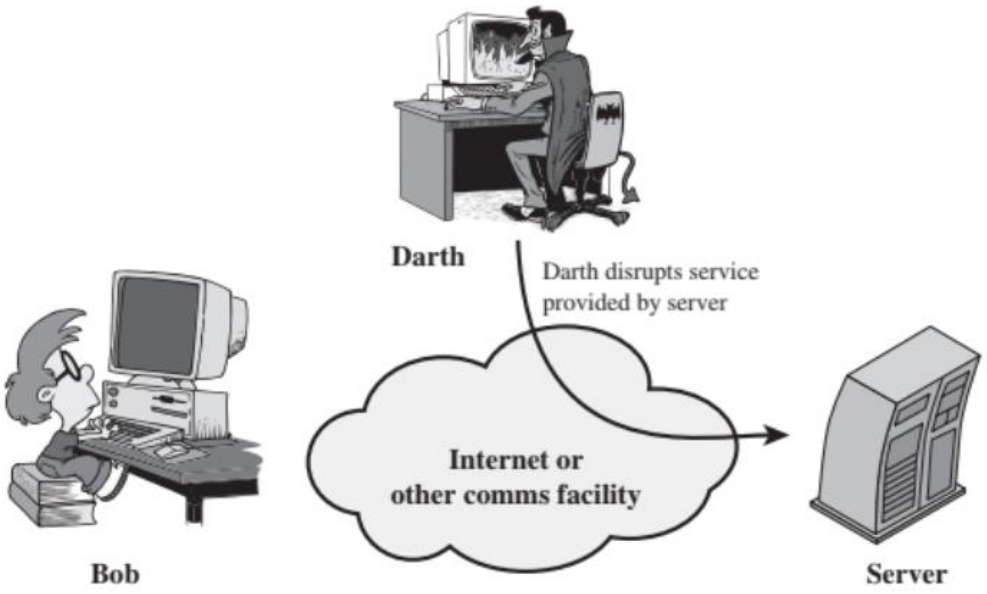
3. Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect For example, a message meaning "Allow John Smith to read confidential file accounts" is modified to mean "Allow Fred Brown to read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



(c) Modification of messages

4. **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.



(d) Denial of service

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.

Security Services

X.800 defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers. X.800 divides these services into five categories and fourteen specific services.

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.

Security Mechanisms

The security mechanisms defined in X.800 are divided into those that are implemented in a specific protocol layer, such as TCP or an application-layer protocol, and those that are not specific to any particular protocol layer or security service.

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.	Mechanisms that are not specific to any particular OSI security service or protocol layer.
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.	**Trusted Functionality** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
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).	**Security Label** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
Access Control A variety of mechanisms that enforce access rights to resources.	**Event Detection** Detection of security-relevant events.
Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.	**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.
Authentication Exchange A mechanism intended to ensure the identity of an entity by means of information exchange.	**Security Recovery** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.
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.	

Chapter No: 02: Classical Encryption Techniques

Before beginning, we define some terms: -

An original message is known as the **plaintext**, while the coded message is called the **ciphertext**. The process of converting from plaintext to ciphertext is known as enciphering or **encryption**; restoring the plain text from the ciphertext is deciphering or **decryption**. The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system** or a cipher. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**. Cryptanalysis is what the layperson calls "breaking the code. The areas of cryptography and cryptanalysis together are called **cryptology**.

Symmetric Cipher Model

A symmetric encryption scheme has five ingredients

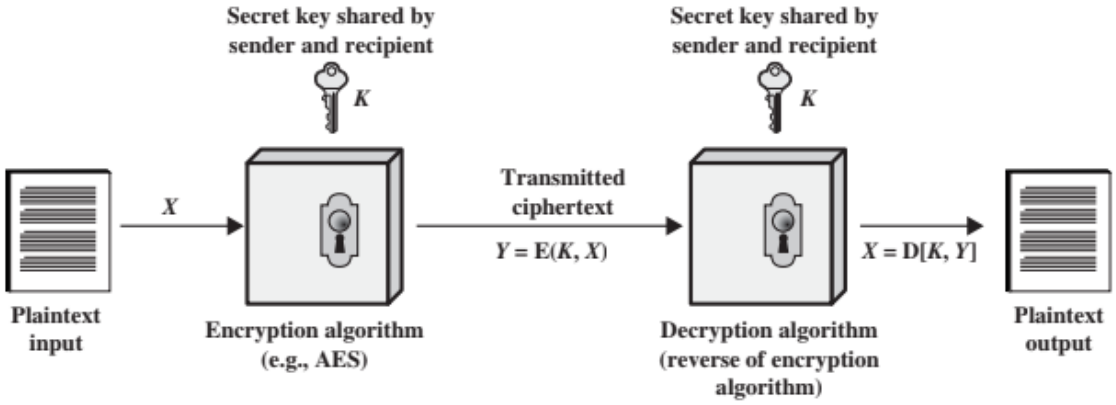
• **Plaintext**: This is the original intelligible message or data that is fed into the algorithm as input.

• **Encryption algorithm**: The encryption algorithm performs various substitutions and transformations on the plaintext.

• **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.

• **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.

• **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.



Cryptography

Cryptographic systems are characterized along three independent dimensions:

1. The type of operations used for transforming plaintext to ciphertext: All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.

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

3. The way in which the plaintext is processed: A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

Substitution Techniques

The two basic building blocks of all encryption techniques are substitution and transposition. A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

Caesar Cipher

The earliest known, and the simplest, use of a substitution cipher was by Julius Caesar. The Caesar cipher method is based on a mono-alphabetic cipher and is also called a shift cipher or additive cipher. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example,

The formula of encryption is: $C = E(k, p) = (p + k) \bmod 26$

The formula of decryption is: $p = D(k, C) = (C - k) \bmod 26$

plain: meet me after the toga party

cipher: PHHW PH DIWHU WKH WRJD SDUWB

Advantages of Caesar cipher

- It is very easy to implement.
- This method is the simplest method of cryptography.
- Only one short key is used in its entire process.
- If a system does not use complex coding techniques, it is the best method for it.
- It requires only a few computing resources.

Disadvantages of Caesar cipher

- It can be easily hacked. It means the message encrypted by this method can be easily decrypted.
- It provides very little security.
- By looking at the pattern of letters in it, the entire message can be decrypted.

Monoalphabetic Ciphers

With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Before proceeding, we define the term permutation. A permutation of a finite set of elements is an ordered sequence of all the elements of, with each element appearing exactly once. Monoalphabetic Cipher is a part of the substitution technique in which a single cipher alphabet is used per message (mapping is done from plain alphabet to cipher alphabet). Monoalphabetic cipher converts plain text into cipher text and re-convert a cipher text to plain text. Monoalphabetic Cipher eliminates the brute-force techniques for cryptanalysis. Moreover, the cipher line can be a permutation of the 26 alphabetic characters.

Example

Input (Plain-text)– GFG Output (Cipher-text)– SRS

Explanation: In Monoalphabetic cipher, the mapping is done randomly and the difference between the letters is not uniform. Here, the word is mapped to S (G->S), F is mapped to R(F->R) and G was already mapped to S so we cannot change it (G->S).

- Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet. A countermeasure is to provide multiple substitutes, known as homophones, for a single letter.
- Prone to guessing attack using the English letters frequency of occurrence of letters.
- The English Language is used so the nature of plain text is known.
- Less secure than a polyalphabetic cipher.

Playfair Cipher

- The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair algorithm is based on the use of a 5 × 5 matrix of letters constructed using a keyword. In this case, the keyword is monarchy. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom,

and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.

Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.

2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.

3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.

4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

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

Advantages:

- It is significantly harder to break since the frequency analysis technique used to break simple substitution ciphers is difficult but still can be used on (25*25) = 625 digraphs rather than 25 monographs which is difficult.
- Frequency analysis thus requires more cipher text to crack the encryption.

Disadvantages:

- An interesting weakness is the fact that a digraph in the ciphertext (AB) and it's reverse (BA) will have corresponding plaintexts like UR and RU (and also ciphertext UR and RU will correspond to plaintext AB and BA, i.e. the

substitution is self-inverse). That can easily be exploited with the aid of frequency analysis, if the language of the plaintext is known.

- Another disadvantage is that playfair cipher is a symmetric cipher thus same key is used for both encryption and decryption.

Hill Cipher

Another interesting multi-letter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

We have to encrypt the message ‘ACT’ (n=3).The key is ‘GYBNQKURP’ which can be written as the nxn matrix: The message ‘ACT’ is written as vector:

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix}$$

The enciphered vector is given as

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} (\text{mod } 26)$$

which corresponds to ciphertext of ‘POH’.

Transposition Techniques

All the techniques examined so far involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

Rail Fence Transposition Cipher

The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message “meet me after the toga party” with a rail fence of depth 2, we write the following: It is also termed as a zigzag cipher. It gets it’s name from the way through which it performs encryption of plain text.

```
m e m a t r h t g p r y
  e t e f e t e o a a t
```

The encrypted message is Ciphertext: **MEMATRHTGPRYETEFETEOAAT**

Block (Single Columnar) Transposition Cipher

This sort of thing would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm. For example,

```
Key:          4 3 1 2 5 6 7
Plaintext:    a t t a c k p
              o s t p o n e
              d u n t i l t
              w o a m x y z
Ciphertext:   TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Thus, in this example, the key is 4312567. To encrypt, start with the column that is labeled 1, in this case column 3. Write down all the letters in that column. Proceed to column 4, which is labeled 2, then column 2, then column 1, then columns 5, 6, and 7.

STEGANOGRAPHY

A plaintext message may be hidden in one of two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text. A simple form of steganography, but one that is time-consuming to construct, is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message.

Various other techniques have been used historically; some examples are the following

Character marking: Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.

Invisible ink: A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

Pin punctures: Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.

Typewriter correction ribbon: Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Sr.NO	Substitution Cipher Technique	Transposition Cipher Technique
1.	In substitution Cipher Technique, plain text characters are replaced with other characters, numbers and symbols.	In transposition Cipher Technique, plain text characters are rearranged with respect to the position.
2.	Substitution Cipher's forms are: Mono alphabetic substitution cipher and poly alphabetic substitution cipher.	Transposition Cipher's forms are: Key-less transposition cipher and keyed transposition cipher.
3.	In substitution Cipher Technique, character's identity is changed while its position remains unchanged.	While in transposition Cipher Technique, The position of the character is changed but character's identity is not changed.
4.	In substitution Cipher Technique, The letter with low frequency can detect plain text.	While in transposition Cipher Technique, The Keys which are nearer to correct key can disclose plain text.
5.	The example of substitution Cipher is Caesar Cipher, monoalphabetic cipher, and polyalphabetic cipher.	The example of transposition Cipher is Rail Fence Cipher, columnar transposition cipher, and route cipher.
6.	Involves replacing plaintext letters or groups of letters with ciphertext letters or groups of letters according to a specific algorithm or key.	Involves rearranging the order of the plaintext letters or groups of letters according to a specific algorithm or key.

Sr.NO	Substitution Cipher Technique	Transposition Cipher Technique
7.	The frequency distribution of the plaintext letters is typically obscured, but patterns can still be detected with statistical analysis.	The frequency distribution of the plaintext letters remains the same, but the order is scrambled, making it difficult to detect patterns with statistical analysis.
8.	Vulnerable to frequency analysis attacks, where the most commonly used letters or letter combinations in the language can be identified and used to deduce the key.	Less vulnerable to frequency analysis attacks, but still susceptible to attacks such as brute force and known plaintext attacks.
9.	Relatively easy to understand and implement, making it suitable for simple applications.	Can be more difficult to implement and understand, but can be more secure than substitution ciphers for certain applications.

Chapter No 03: Block Ciphers and The Data Encryption Standard

A block cipher is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used.

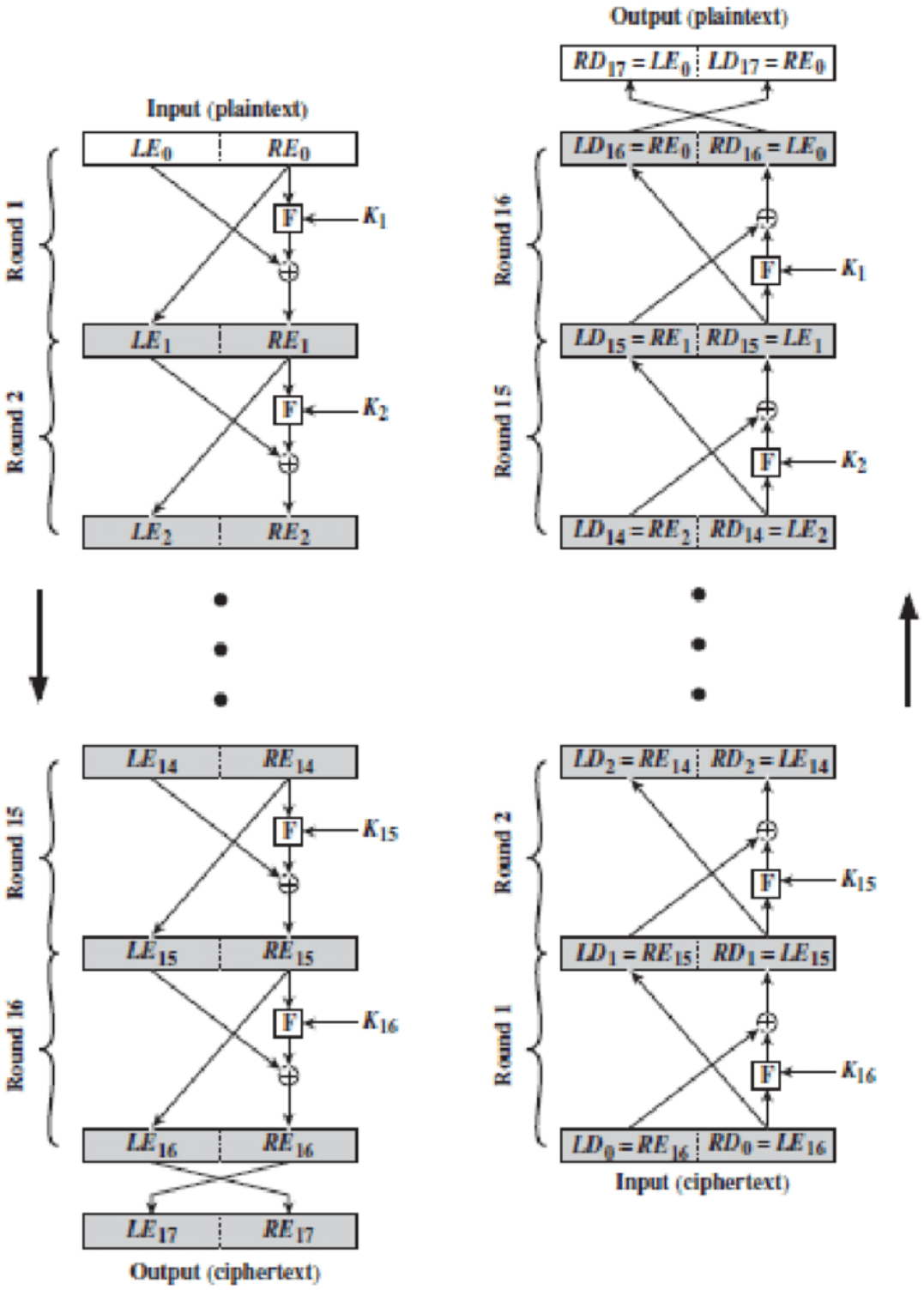
The Feistel Cipher

Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of k bits and a block length of bits, allowing a total of possible transformations, rather than the transformations available with the ideal block cipher. In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:

• **Substitution**: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.

• **Permutation**: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

- The left-hand side of Figure depicts the structure proposed by Feistel.
- The inputs to the encryption algorithm are a plaintext block of length bits and a key . The plaintext block is divided into two halves.
- The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block.
- Each round has as inputs and derived from the previous round, as well as a subkey derived from the overall.
- In general, the subkeys are different from and from each other. 16 rounds are used, although any number of rounds could be implemented. All rounds have the same structure.



- A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey.

- Another way to express this is to say that F is a function of right-half block of bits and a subkey of bits, which produces an output value of length w bits: Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

• **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

• **Key size:** Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.

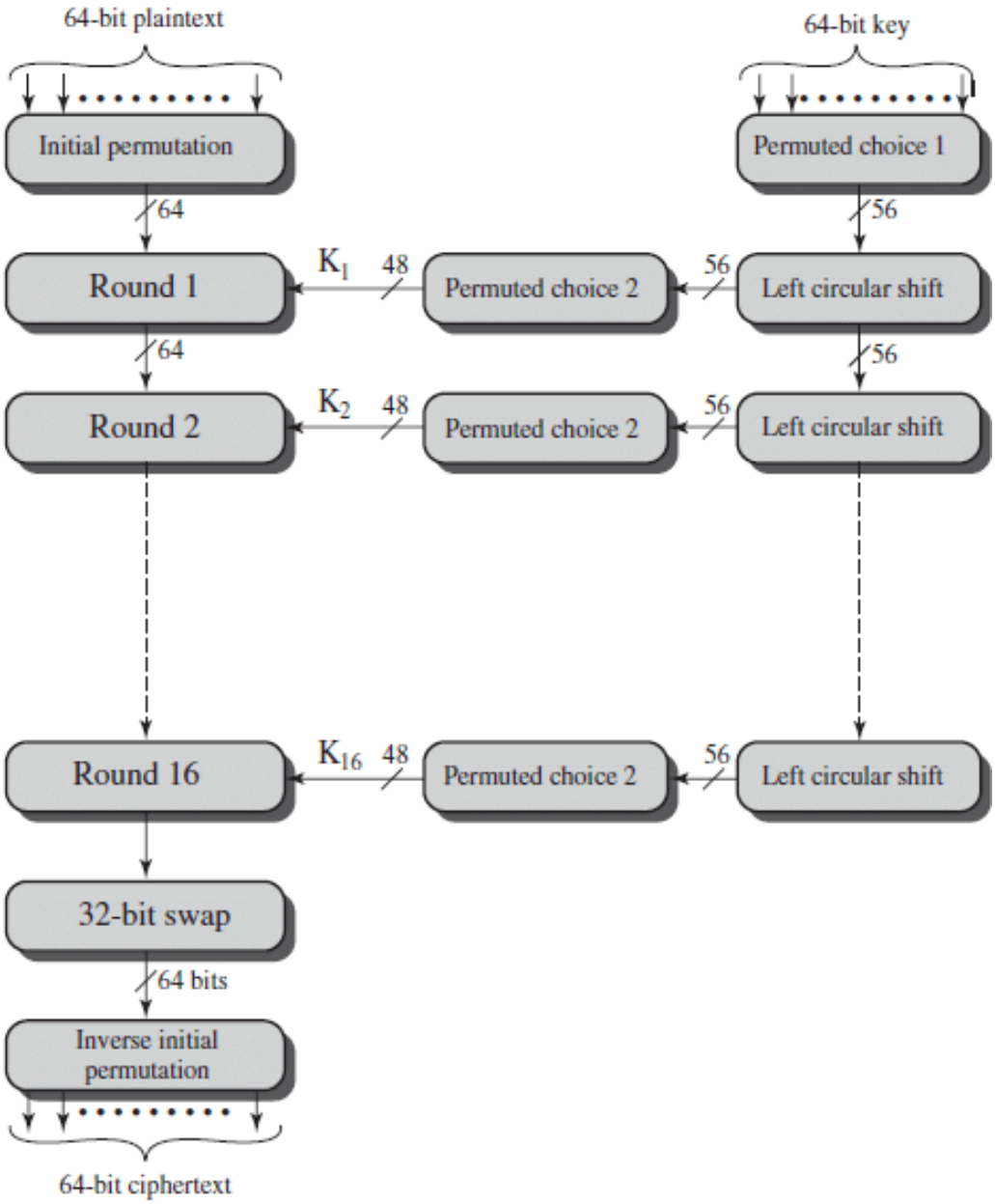
• **Number of rounds**: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

• **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

• **Round function F:** Again, greater complexity generally means greater resistance to cryptanalysis.

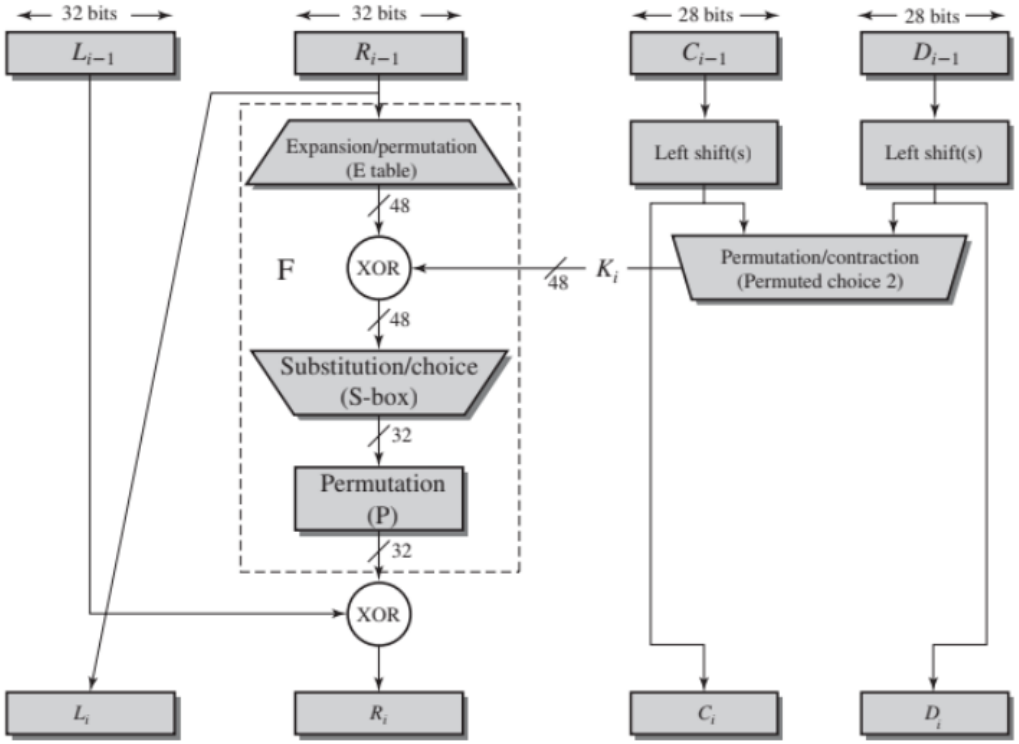
Data Encryption Standard (DES)

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption. The overall scheme for DES encryption is illustrated As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.



Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.

- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
- This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions.
- The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key.
- The left and right halves of the output are swapped to produce the pre-output. Finally, the pre-output is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.
- With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.
- The right-hand portion shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function.
- Then, for each of the sixteen rounds, a subkey ( ) is produced by the combination of a left. circular shift and a permutation.
- The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.



The Strength Of DES

There are mainly two categories of concerns about the strength of Data encryption standard. They are:

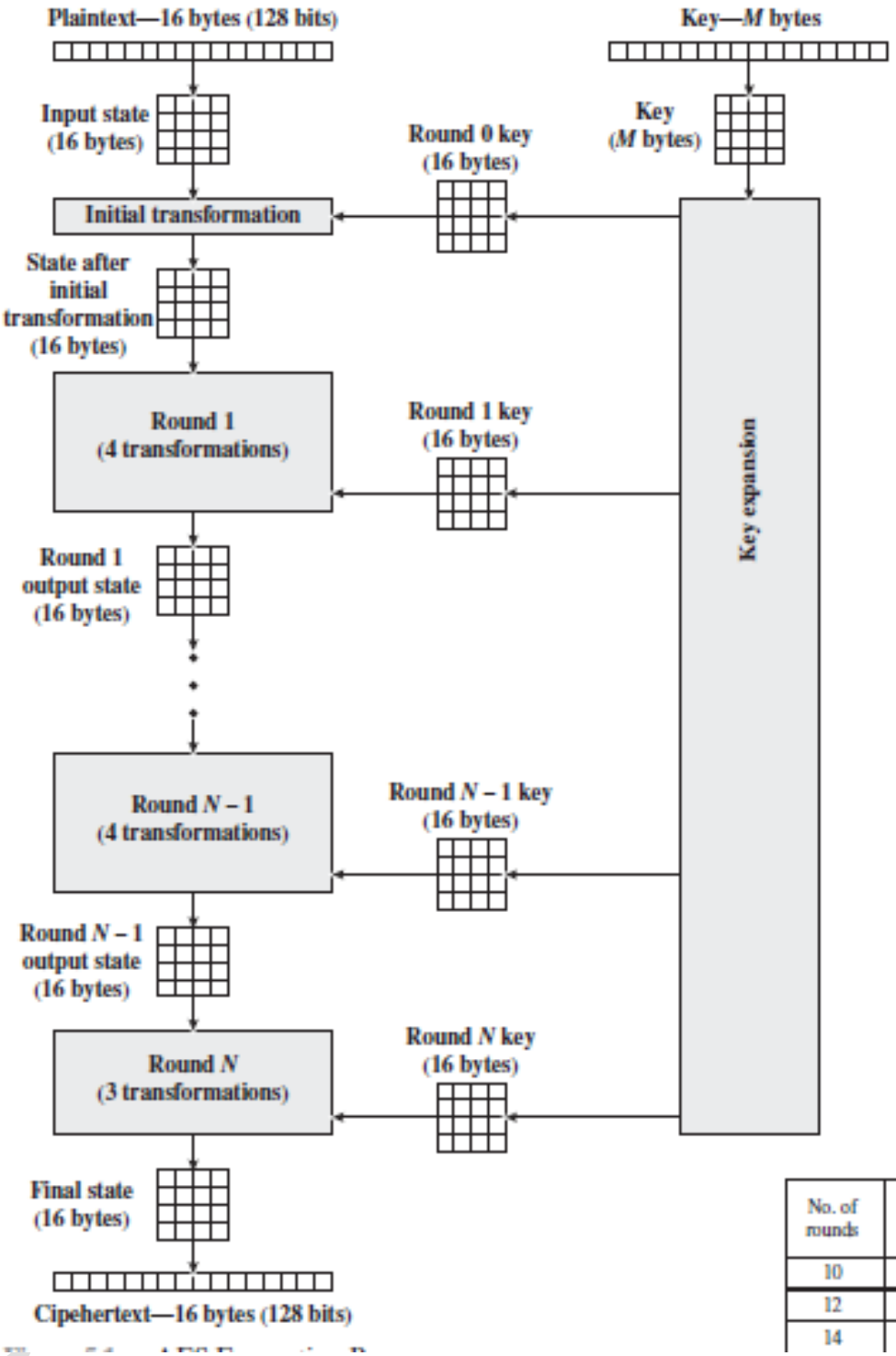
1. Concerns about the particular algorithm used.
2. Concerns about the usage of key of size 56-bit.

The first concern regarding the algorithm used addresses the possibility of cryptanalysis by making use of the DES algorithm characteristics. A more severe concern is about the length of secret key used. There can be (approximately 7.2 × keys) possible keys with a key length of 56 bits. Thus, a brute force attack appears to be impractical. Assuming that on an average one has to search half the key space, to break the cipher text, a system performing one DES encryption per microsecond might require more than thousand years. But, the assumption of one DES encryption per microsecond is too conservative. In July 1998, DES was finally proved to be insecure when the Electronic Frontier Foundation (EFF) had broken a DES encryption. The encryption was broken with the help of a special-purpose “DES cracker” machine. It was reported that the attack took less than 3 days. Simply running through all possible keys won’t result in cracking the DES encryption. Unless known plain text is given, the attacker must be able to differentiate the plain text from other data. Some degree of knowledge about the target plain text and some techniques for automatically distinguishing plain text from garble are required to supplement the brute-force approach. If brute force attack is the only means to crack the DES encryption algorithm, then using longer keys will obviously help us to counter such

attacks. An algorithm is guaranteed unbreakable by brute force if a 128- bit key is used. The differential cryptanalysis, linear cryptanalysis, are examples for statistical attacks on DES algorithm. Few of the important alternatives for DES are AES (Advanced Encryption Standard) and triple DES.

AES STRUCTURE

- The cipher takes a plaintext block size of 128 bits, or 16 bytes. 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.
- The input to the encryption and decryption algorithms is a single 128-bit block, this block is depicted as a 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. These operations are depicted in Figure below.
- Similarly, the key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words. Figure below shows the expansion for the 128-bit key. Each word is four bytes, and the total key schedule is 44 words for the 128-bit key.
- Note that the ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the in matrix, the second four bytes occupy the second column, and so on.
- Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the w matrix.
- The cipher consists of rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key.
- The first rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are described subsequently.
- The final round contains only three transformations, and there is a initial single transformation (AddRoundKey) before the first round, which can be considered Round 0.
- Each transformation takes one or more matrices as input and produces a matrix as output. Figure below shows that the output of each round is a matrix, with the output of the final round being the ciphertext. Also, the key expansion function generates round keys, each of which is a distinct matrix.
- Each round key serve as one of the inputs to the AddRoundKey transformation in each round.

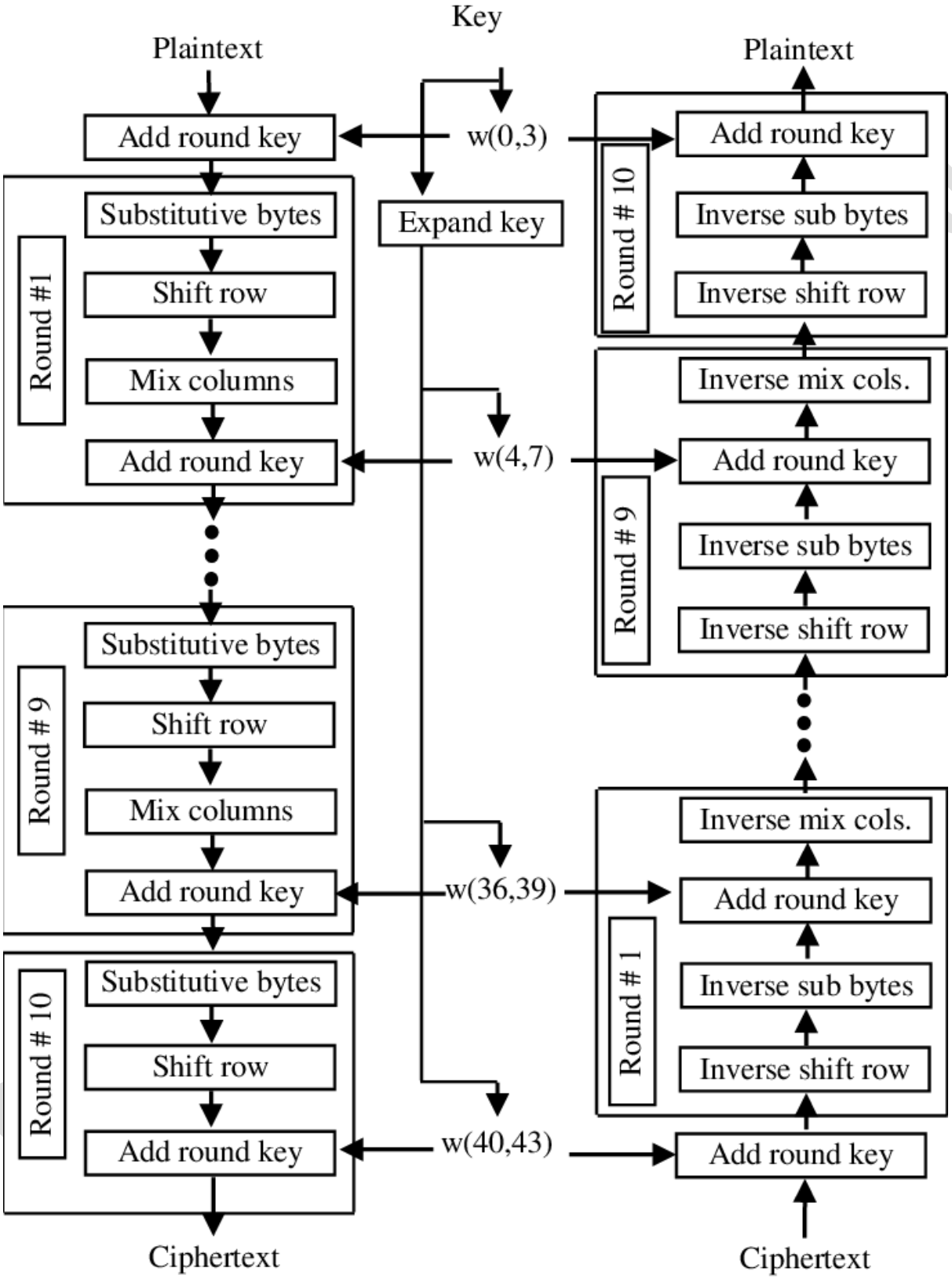


No. of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Parameters

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

Detailed Structure of AES



- One noteworthy feature of this structure is that it is not a Feistel structure. Recall that, in the classic Feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped. AES instead 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; these are indicated in Figure above.
- 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 over

• **AddRoundKey**: A simple bitwise XOR of the current block with a portion of the expanded key

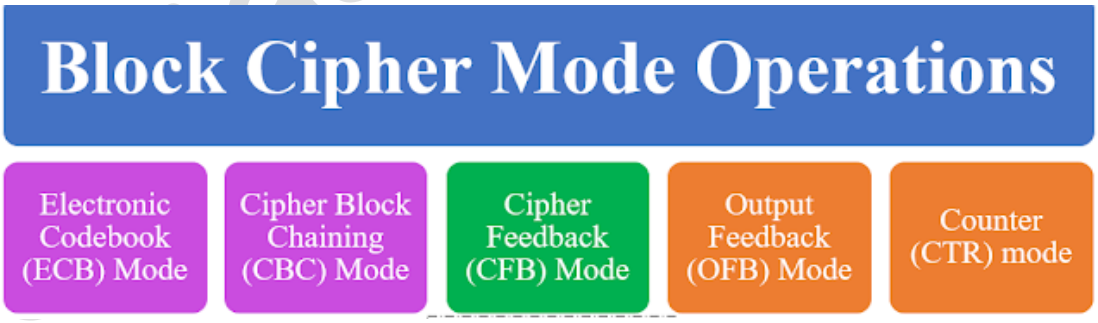
- The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages. Figure above depicts the structure of a full encryption round.
- Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.
- The AddRoundKey stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on. This scheme is both efficient and highly secure.
- Each stage is easily reversible. For the Substitute Byte, ShiftRows, and MixColumns stages, an inverse function is used in the decryption algorithm. For the AddRoundKey

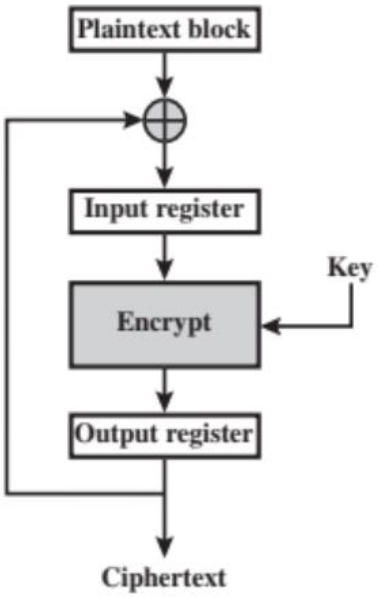
stage, the inverse is achieved by XORing the same round key to the block, using the result that $A \oplus B \oplus B = A$

- As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm. This is a consequence of the particular structure of AES.
- Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext. Figure lays out encryption and decryption going in opposite vertical directions. At each horizontal point (e.g., the dashed line in the figure), State is the same for both encryption and decryption.
- The final round of both encryption and decryption consists of only three stages. Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible.

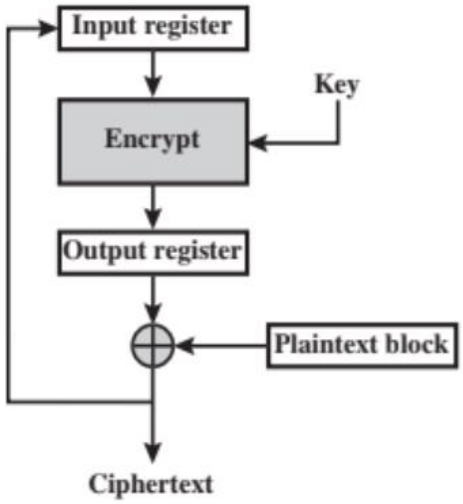
Block Cipher Modes of Operation

- A block cipher takes a fixed-length block of text of length bits and a key as input and produces a -bit block of ciphertext. If the amount of plaintext to be encrypted is greater than bits, then the block cipher can still be used by breaking the plaintext. up into -bit blocks. When multiple blocks of plaintext are encrypted using the same key, a number of security issues arise. To apply a block cipher in a variety of applications, five modes of operation have been defined by NIST (SP 800-38A).

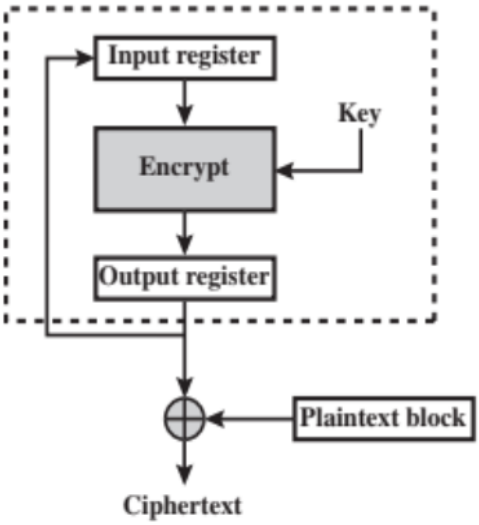




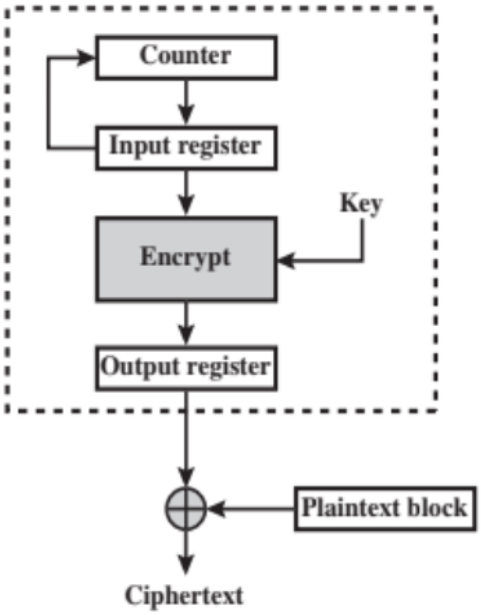
(a) Cipher block chaining (CBC) mode



(b) Cipher feedback (CFB) mode

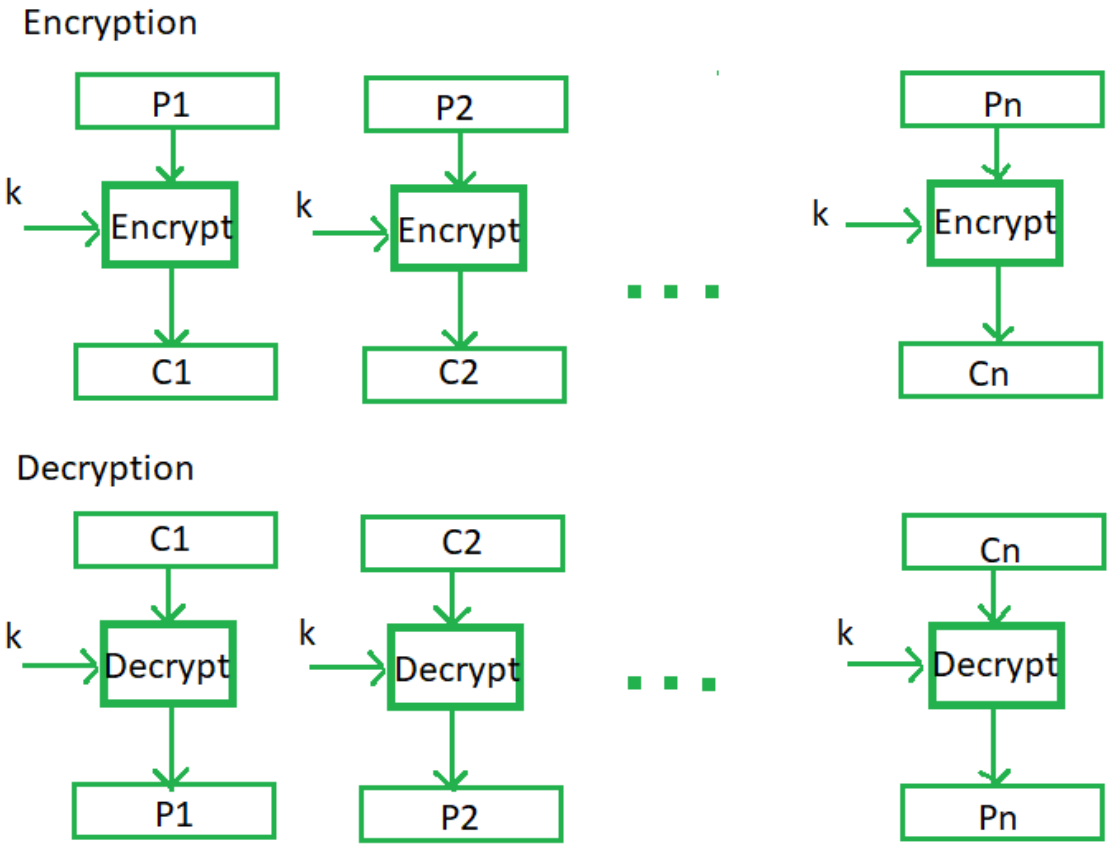


(c) Output feedback (OFB) mode



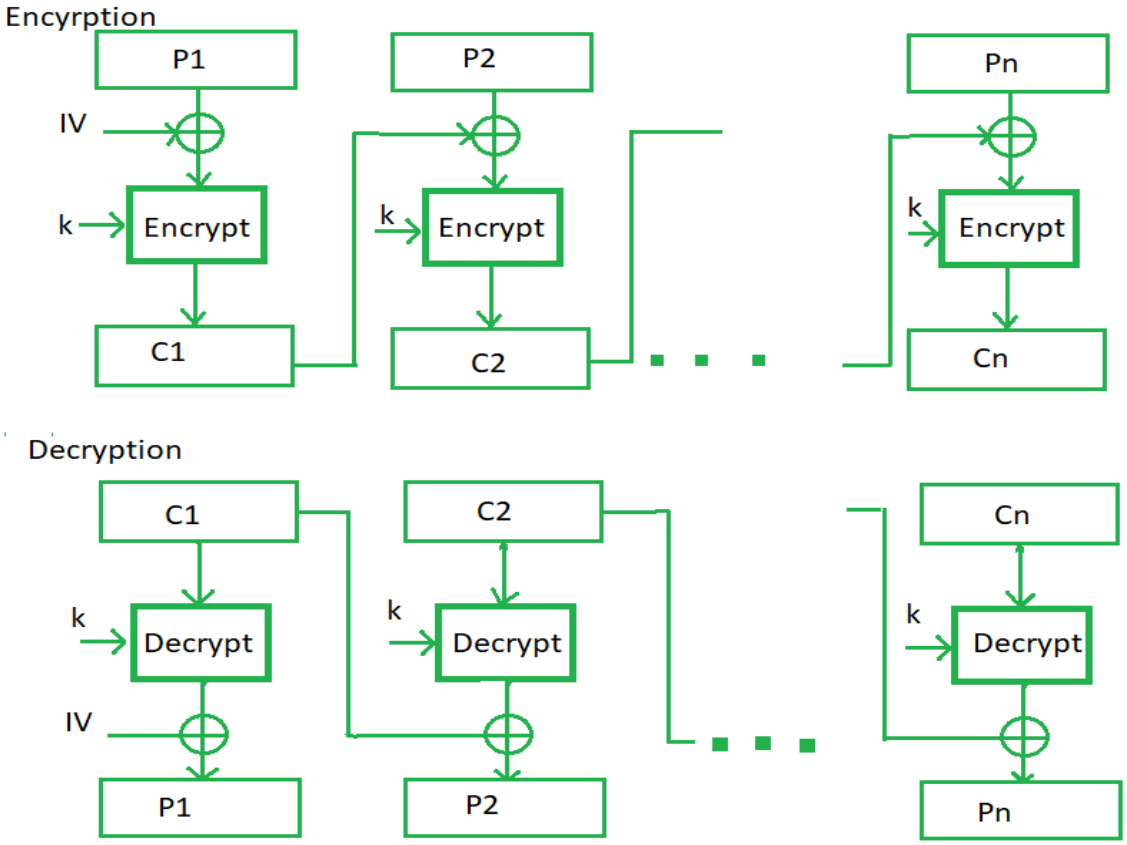
(d) Counter (CTR) mode

Electronic codebook (ECB)



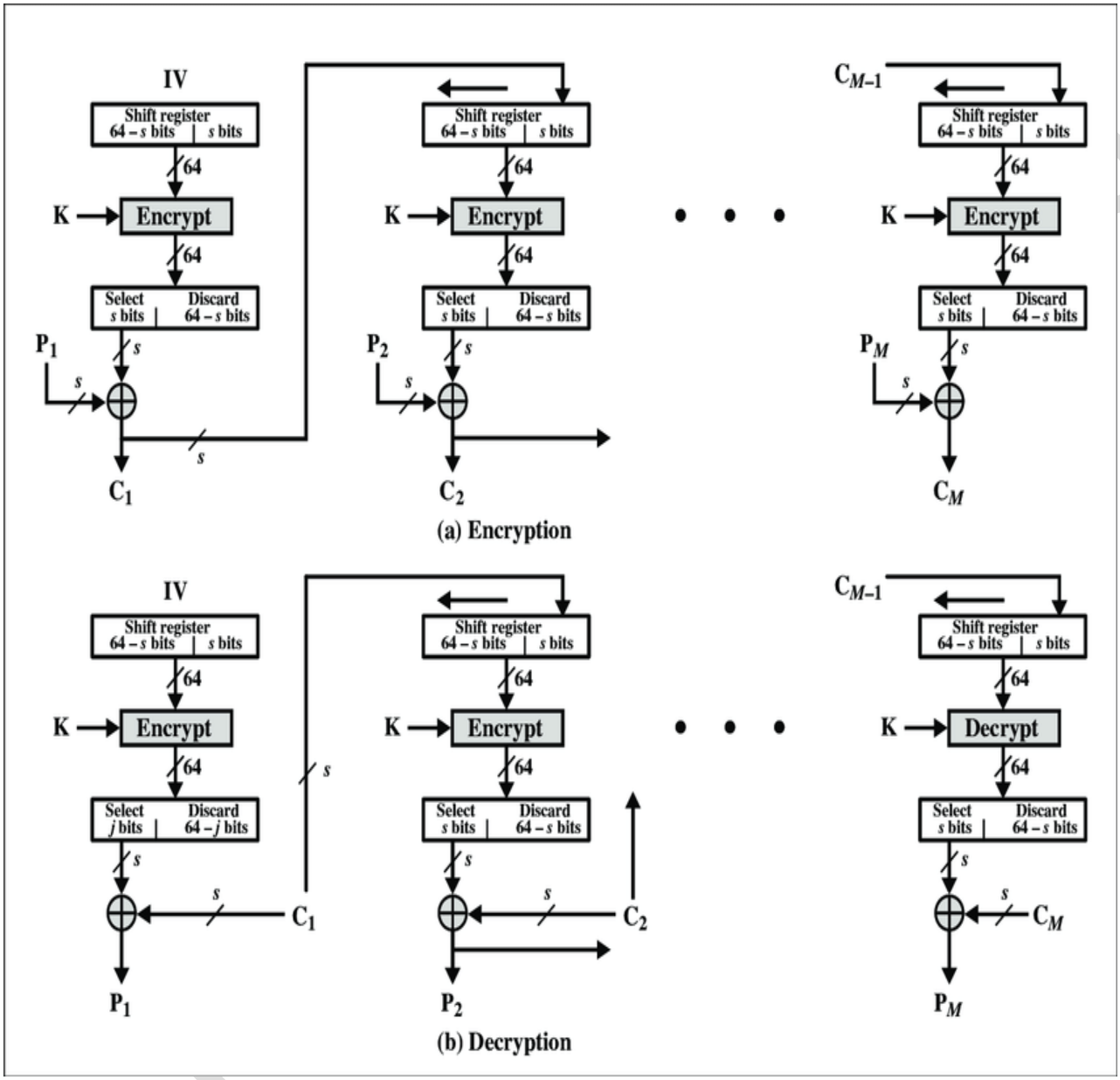
- The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key. The term codebook is used because, for a given key, there is a unique ciphertext for every -bit block of plaintext. Therefore, we can imagine a gigantic codebook in which there is an entry for every possible bit plaintext pattern showing its corresponding ciphertext.
- For a message longer than bits, the procedure is simply to break the message into -bit blocks, padding the last block if necessary. Decryption is performed one block at a time, always using the same key.
- The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES or AES key securely, ECB is the appropriate mode to use. The most significant characteristic of ECB is that if the same -bit block of plaintext appears more than once in the message, it always produces the same ciphertext.
- For lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities. For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintexts–ciphertext pairs to work with.
- If the message has repetitive elements with a period of repetition a multiple of bits, then these elements can be identified by the analyst. This may help in the analysis or may provide an opportunity for substituting or rearranging blocks.

Cipher Block Chaining Mode



- To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher block chaining (CBC) mode.
- In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks.
- The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of bits are not exposed. As with the ECB mode, the CBC mode requires that the last block be padded to a full bits if it is a partial block.
- For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block.

Cipher Feedback Mode



- For AES, DES, or any block cipher, encryption is performed on a block of bits. In the case of DES, and in the case of AES, However, it is possible to convert a block cipher into a stream cipher, using one of the three modes to be discussed in this and the next two sections: cipher feedback (CFB) mode, output feedback (OFB) mode, and counter (CTR) mode.
- A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

- One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a ciphertext output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted.
- First, consider encryption. The input to the encryption function is a -bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) bits of the output of the encryption function are XORed with the first segment of plaintext to produce the first unit of ciphertext, which is then transmitted. In addition, the contents of the shift register are shifted left by bits, and is placed in the rightmost (least significant) bits of the shift register. This process continues until all plaintext units have been encrypted.
- For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the encryption function that is used, not the decryption function. This is easily explained.

Output Feedback Mode

- The output feedback (OFB) mode is similar in structure to that of CFB. As can be seen, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB, the ciphertext unit is fed back to the shift register. The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on s-bit subset.

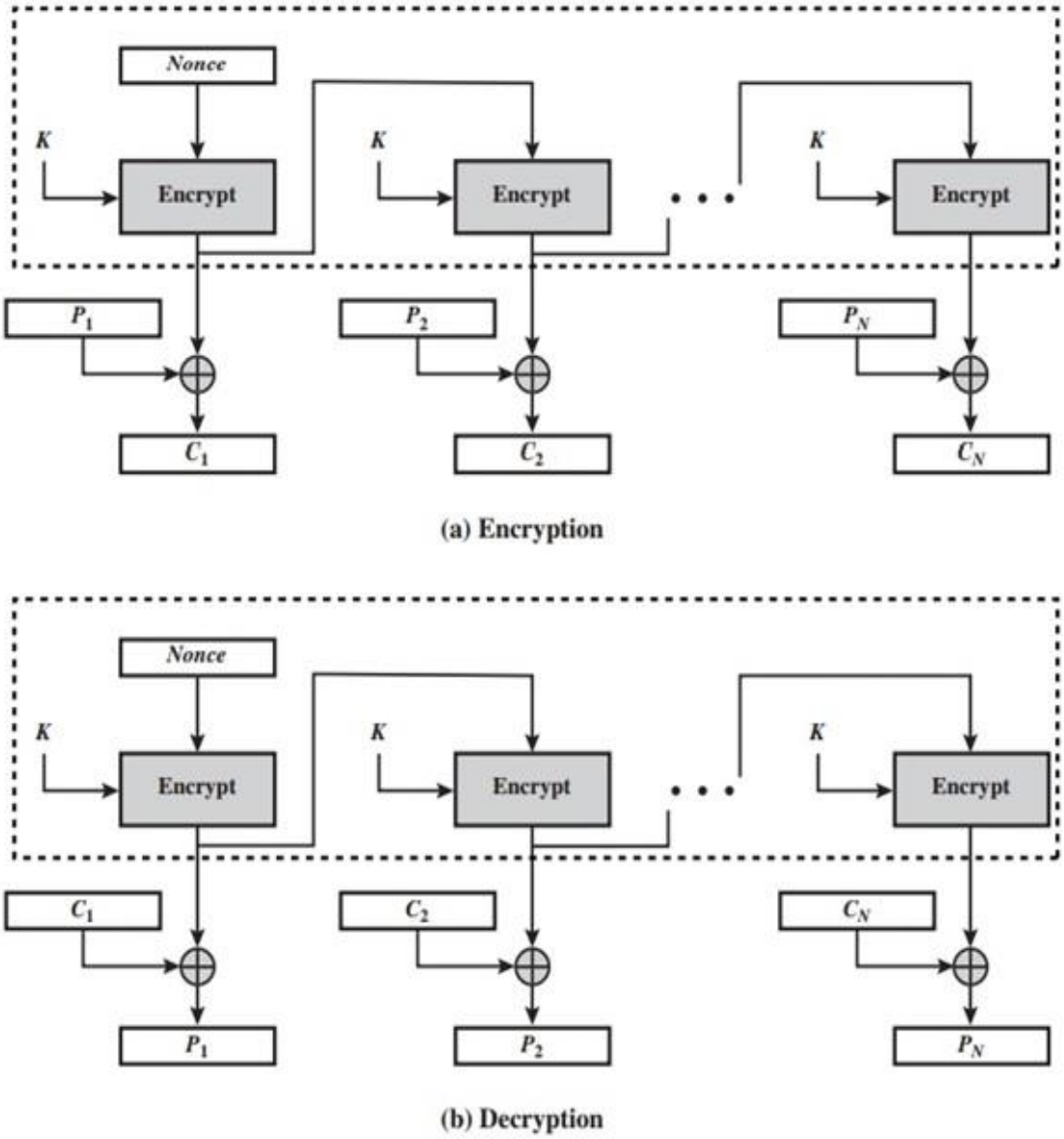


Figure 6.6 Output Feedback (OFB) Mode

- As with CBC and CFB, the OFB mode requires an initialization vector. In the case of OFB, the IV must be a nonce; that is, the IV must be unique to each execution of the encryption operation. The reason for this is that the sequence of encryption output blocks, , depends only on the key and the IV and does not depend on the plaintext. Therefore, for a given key and IV, the stream of output bits used to XOR with the stream of plaintext bits is fixed. If two different messages had an identical block of plaintext in the identical position, then an attacker would be able to determine that portion of the stream.

- One advantage of the OFB method is that bit errors in transmission do not propagate. For example, if a bit error occurs in, only the recovered value of is affected; subsequent plaintext units are not corrupted. With CFB, also serves as input to the shift register and therefore causes additional corruption downstream.

- The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB. Consider that complementing a bit in the cipher text complements the corresponding bit in the recovered plaintext. Thus, controlled changes to the recovered plaintext can be made. This may make it possible for an opponent, by making the necessary changes to the checksum portion of the message as well as to the data portion, to alter the ciphertext in such a way that it is not detected by an error-correcting code.

Counter Mode

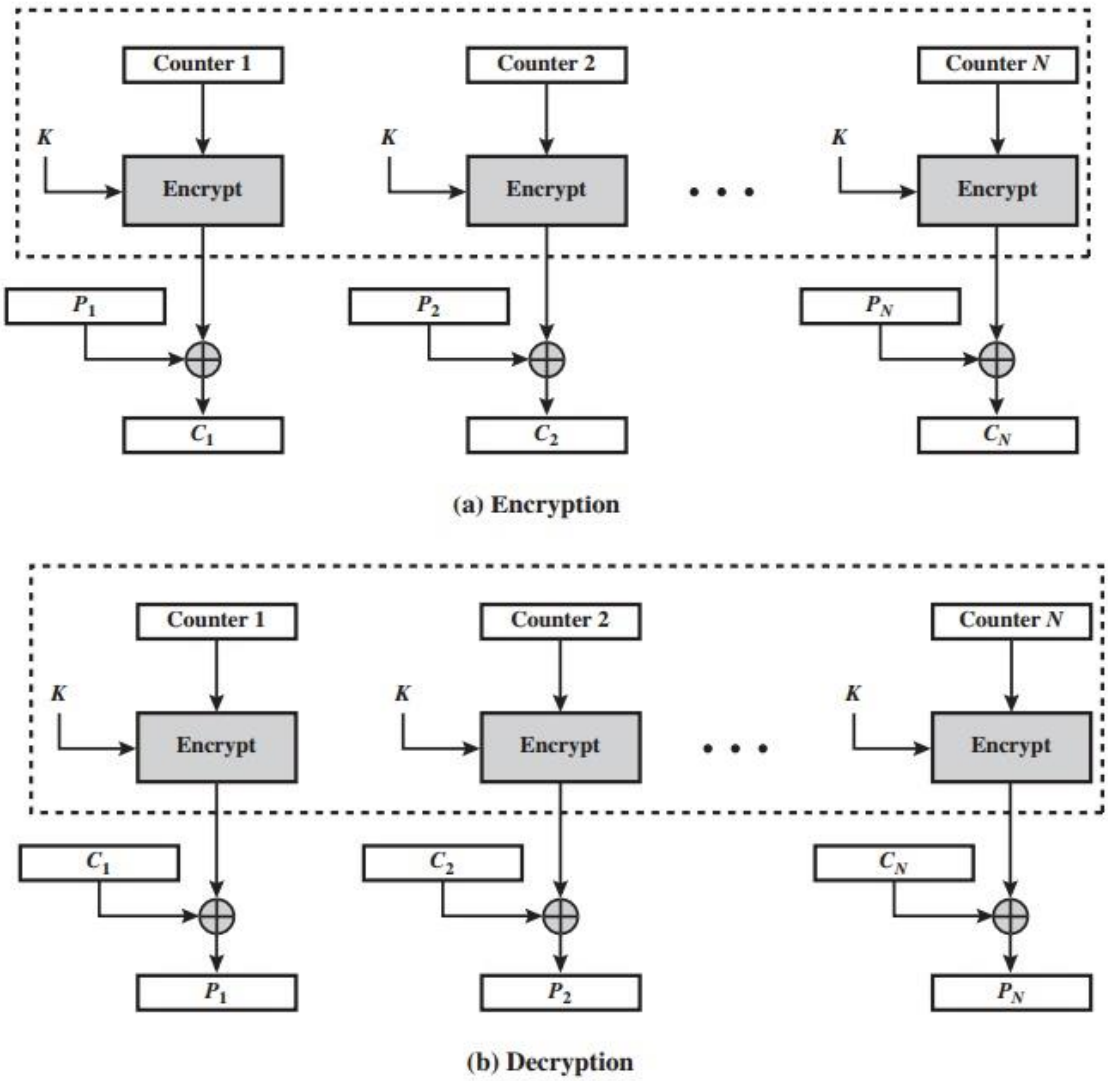


Figure 6.7 Counter (CTR) Mode

- Although interest in the counter (CTR) mode has increased recently with applications to ATM (asynchronous transfer mode) network security and IP sec (IP security), this mode was proposed early on (e.g., [DIFF79]). Figure below depicts the CTR mode. A counter equal to the plaintext block size is used. The only requirement stated in SP 800-38A is that the counter value must be different for each plaintext block that is encrypted.

- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo, where is the block size). For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with

each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block. Thus, the initial counter value must be made available for decryption.

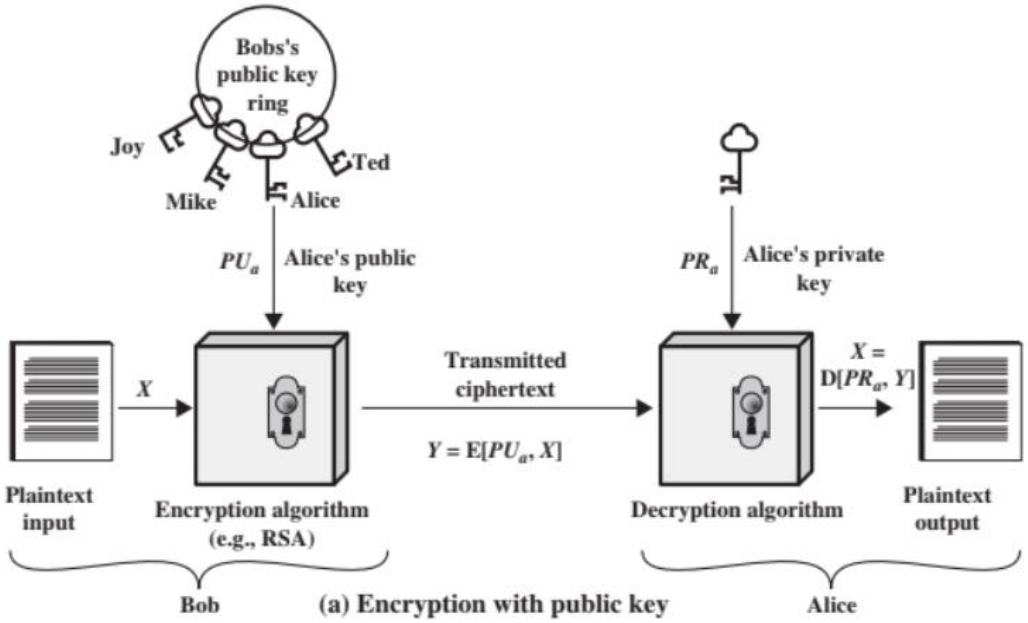
lists the following advantages of CTR mode.

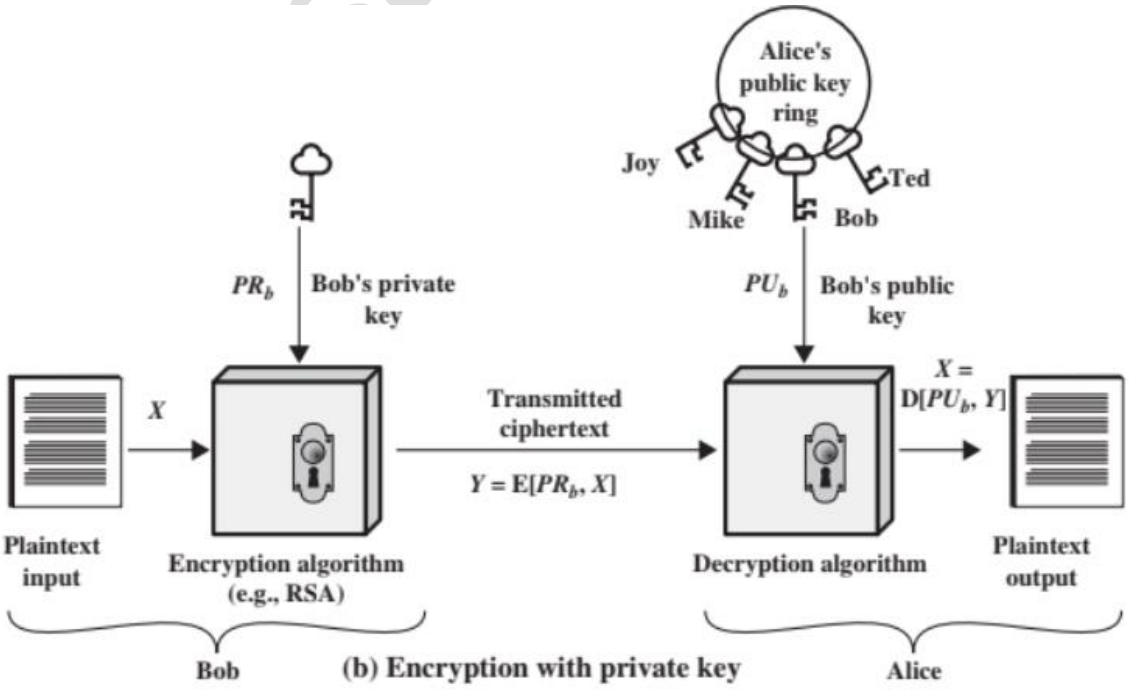
- **Hardware efficiency:** Unlike the three chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plain text or ciphertext. For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block. This limits the maximum throughput of the algorithm to the reciprocal of the time for one execution of block encryption or decryption. In CTR mode, the throughput is only limited by the amount of parallelism that is achieved.
- **Software efficiency:** Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions, can be effectively utilized.
- **Preprocessing:** The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext. Therefore, if sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions, When the plaintext or ciphertext input is presented, then the only computation is a series of XORs. Such a strategy greatly enhances throughput.
- **Random access:** The *ith* block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block cannot be computed until the – 1 prior block are computed. There may be applications in which a ciphertext is stored and it is desired to decrypt just one block; for such applications, the random-access feature is attractive.
- **Provable security:** It can be shown that CTR is at least as secure as the other modes discussed in this section.
- **Simplicity:** Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm. This matters most when the decryption algorithm differs substantially from the encryption algorithm, as it does for AES. In addition, the decryption key scheduling need not be implemented.

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

PUBLIC-KEY CRYPTOGRAPHY

- 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. 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.
- **Ciphertext:** 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 ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.

2. 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, each user maintains a collection of public keys obtained from others.

3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.

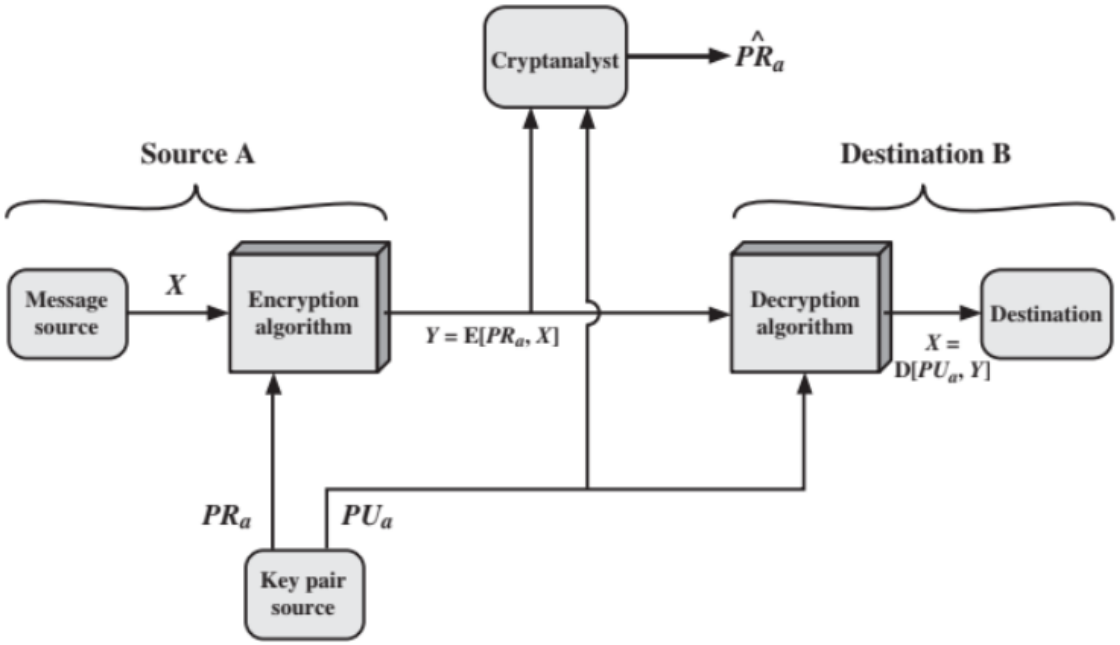
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

5. With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

In broad terms, we can classify the use of public-key cryptosystems 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.



- In this case, A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message.
- Therefore, the entire encrypted message serves as a digital signature. In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

RSA Algorithm

- Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.
- 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}.
- Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to $\log_2(n) + 1$; in practice, the block size is *i* bits, where $2^i < n \leq 2^{i+1}$.
- Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C.

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

- 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}.
- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.

2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.

3. It is infeasible to determine d given e and n.

- For now, we focus on the first requirement and consider the other questions later. We need to find a relationship of the form

$M^{ed} \bmod n = M$

- The preceding relationship holds if *e* and *d,* are multiplicative inverses modulo φ(n), where φ(n) is the Euler totient function.

- p, q prime are two prime numbers (private, chosen),
- n = pq (public, calculated)
- φ (pq) = (p - 1) (q - 1)
- ***e, with gcd (φ (n), e) = 1;*** 1 < e < f(n) (public, chosen)
- The relationship between e and d can be expressed as ***ed mod φ(n) = 1***
- $d = e^{-1} \pmod{\varphi(n)}$ (private, calculated)
- That is, e and d are multiplicative inverses mod f(n). Note that, according to the rules

of modular arithmetic, this is true only if d (and therefore e) is relatively prime to f(n). Equivalently, *gcd(f(n), d) = 1.*

- 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 $\boldsymbol{C = M^e \bmod n}$ and transmits C. On receipt of this ciphertext, user A decrypts by calculating $\boldsymbol{M = C^d \bmod n}$.
- Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key. An example:

For this example, the keys were generated as follows.

1. Select two prime numbers, p = 17 and q = 11.

2. Calculate n = pq = 17 × 11 = 187.

3. Calculate ***φ*** (n) = (p - 1) (q - 1) = 16 × 10 = 160.

4. Select e such that e is relatively prime to ***φ*** (n) = 160 and less than ***φ*** (n); we choose e = 7.

5. Determine d such that de=1 (mod 160) and d < 160.The correct value is d = 23, because

23 × 7 = 161 = (1 × 160) + 1

6. Can be calculated using the extended Euclid's algorithm.

7. The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}.

8. The example shows the use of these keys for a plaintext input of M = 88.

9. For encryption, we need to calculate C = 887 mod 187.

Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

Key Generation

Select p, q	p and q both prime
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext	$M < n$
Ciphertext	$C = M^e \pmod n$

Decryption

Ciphertext	C
Plaintext	$M = C^d \pmod n$

Diffie-Hellman Key Exchange

- The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76b] and is generally referred to as 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 encryption of messages. The algorithm itself is limited to the exchange of secret values.
- 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 exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$.
- Similarly, user B independently selects a random integer $\mathbf{X_B < q}$ and computes

$Y_B = \alpha^{X_B} \bmod q$.

- Each side keeps the value private and makes the value available publicly to the other side. User A and user B computes the key as,

User A = $K = (Y_B)^{XA} \bmod q$ User B = $K = (Y_A)^{XB} \bmod q$

- The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.
- Here is an example. Key exchange is based on the use of the prime number **q=353** and a primitive root of 353, in this case ***α=3***. A and B select secret keys $X_A = 97$ and $X_B = 233$ respectively.
- Now each of them compute their 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 $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{XA} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{XB} \bmod q$

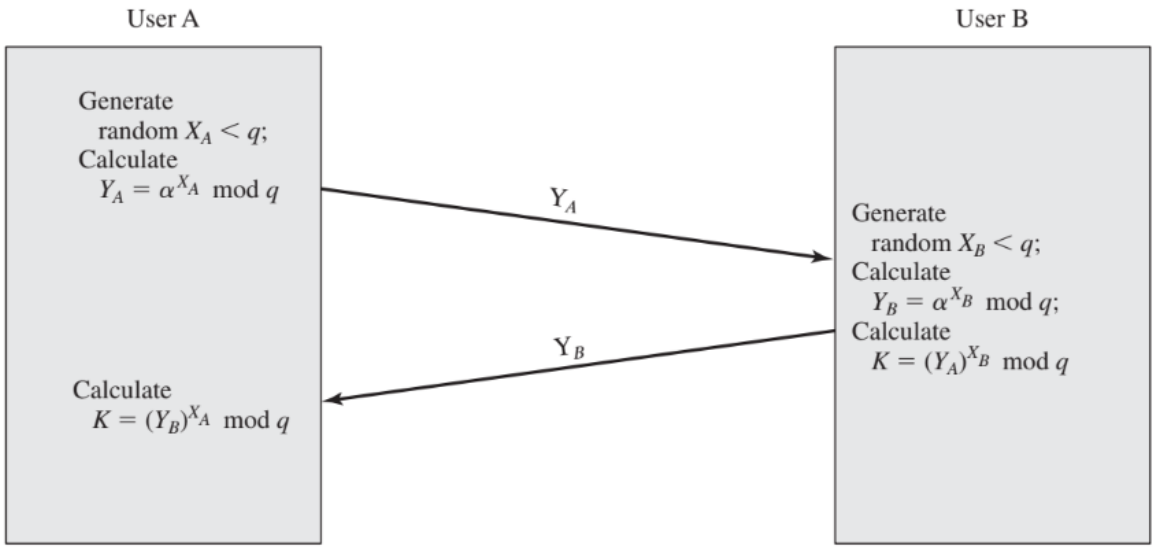


Figure 10.2 Diffie-Hellman Key Exchange

Cryptographic Hash Functions

- A **hash function** H accepts a variable-length block of data as input and produces a fixed-size hash value.
- A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random.
- In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in results, with high probability, in a change to the hash code.
- The kind of hash function needed for security applications is referred to as a **cryptographic hash function.**
- A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either

(a) a data object that maps to a pre-specified hash result (the one-way property) or

(b) two data objects that map to the same hash result (the collision-free property). Because of these characteristics, hash functions are often used to determine whether or not data has changed.

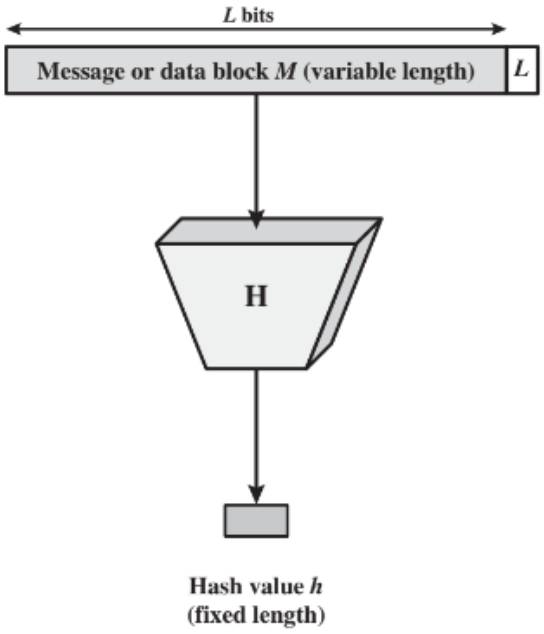


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

Applications Of Cryptographic Hash Functions

Message Authentication

- Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).
- In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid. When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

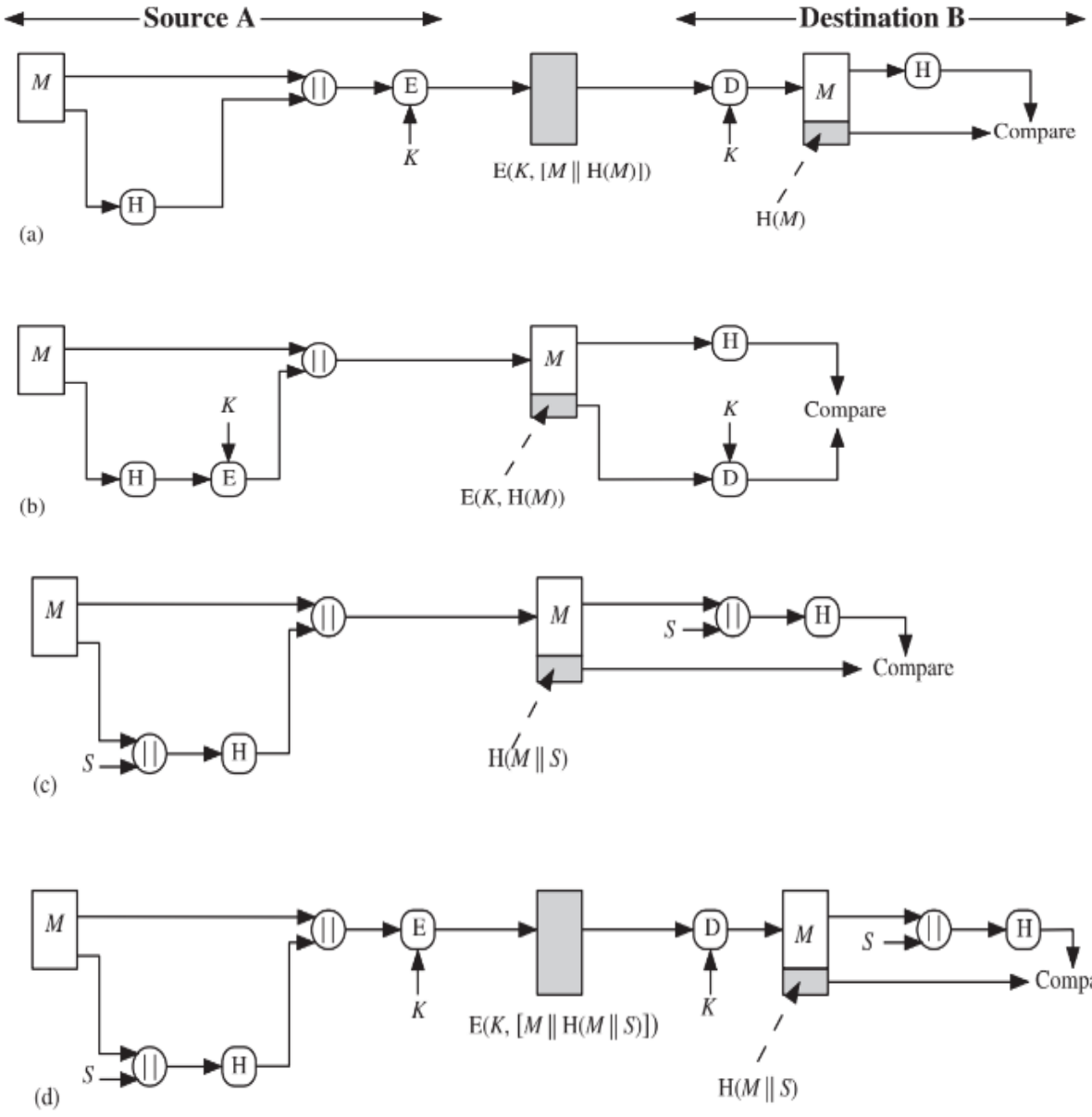


Fig: Simplified Examples of the Use of a Hash Function for Message Authentication

a) The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c) It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value. A compute the hash value over the concatenation of *M* and *S* appends the resulting hash value to *M*. Because B possesses S, it can recompute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

When confidentiality is not required, method (b) has an advantage over methods (a) and (d), which encrypts the entire message, in that less computation is required.

- More commonly, message authentication is achieved using a **message authentication code (MAC)**, also known as a **keyed hash function**. Typically, **MACs** are used between two parties that share a secret key to authenticate information exchanged between those parties.
- A MAC function takes as input a secret key and a data block and produces a hash value, referred to as the MAC. This can then be transmitted with or stored with the protected message.
- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the stored MAC value.
- An attacker who alters the message will be unable to alter the MAC value without knowledge of the secret key. Note that the verifying party also knows who the sending party is because no one else knows the secret key.

Secure Hash Algorithm (SHA)

- In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA). Indeed, because virtually every other widely used hash function had been found to have substantial cryptanalytic weaknesses,
- SHA was more or less the last remaining standardized hash algorithm by 2005. SHA was developed by the National Institute of Standards and Technology (**NIST**) and published as a federal information processing standard (FIPS 180) in 1993.
- When weaknesses were discovered in SHA, now known as **SHA-0**, a revised version was issued as FIPS 180-1 in 1995 and is referred to as **SHA-1.**
- The actual standards document is entitled "Secure Hash Standard." SHA is based on the hash function **MD4**, and its design closely models MD4. SHA-1 is also specified in RFC 3174.
- **SHA-1** produces a hash value of **160** bits. In 2002, NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as **SHA-256, SHA-384**, and **SHA-512**, respectively. Collectively, these hash algorithms are known as SHA-2.
- In 2005, NIST announced the intention to phase out approval of SHA-1 and move to a reliance on SHA-2 by 2010.

SHA-512 Logic

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

Table 11.3 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.

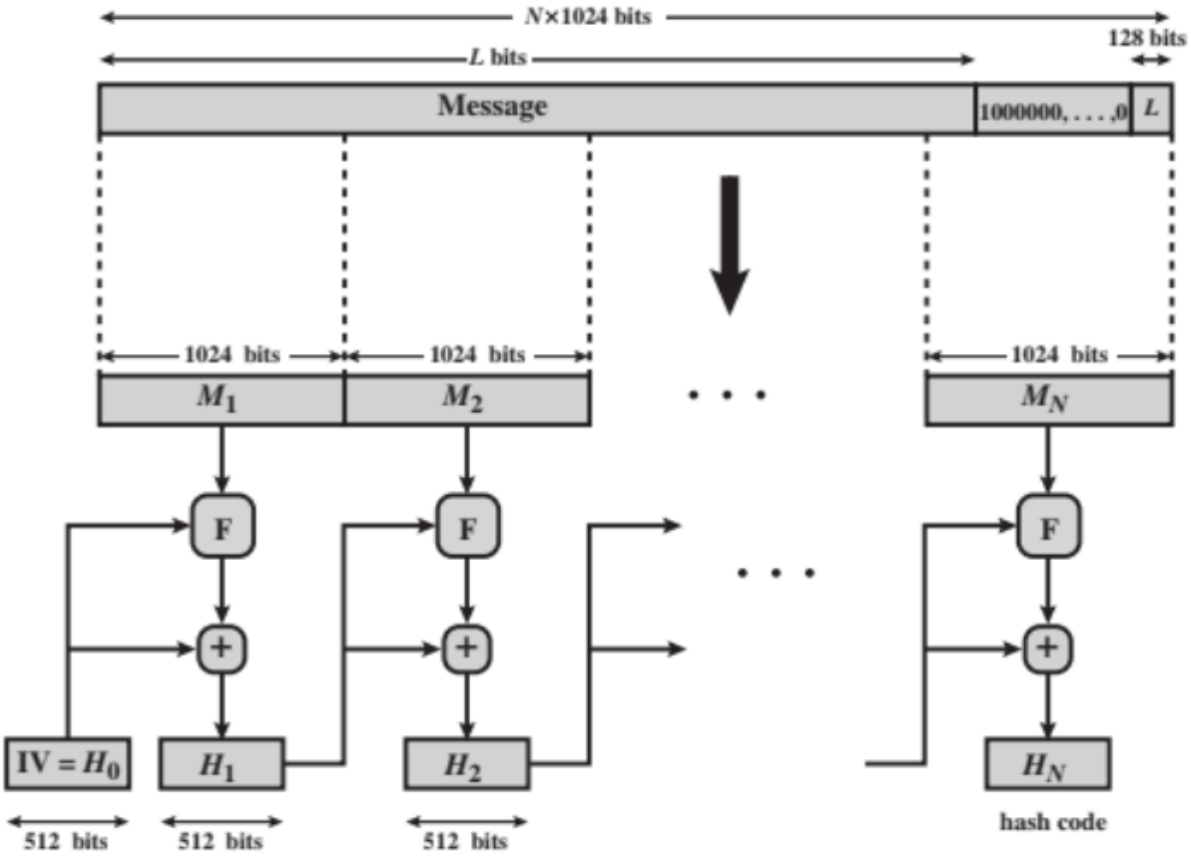


Figure: Message Digest Generation Using SHA-512

the general structure depicted in Figure given above. The processing consists of the following steps.

Step 1: Append padding bits. The message is padded so that its length is congruent to 896 modulo 1024. **[length = 896(mod 1024)]** Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step 2: Append length. A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding). The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In the above Figure, the expanded message is represented as the sequence of 1024-bit blocks *M1, M2, ... Mn*, so that the total length of the expanded message is N X 1024 bits.

Step 3: Initialize hash buffer. A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized to the following 64-bit integers (hexadecimal values):

```
a = 6A09E667F3BCC908   e = 510E527FADE682D1
b = BB67AE8584CAA73B   f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B   g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1   h = 5BE0CD19137E2179
```

These values are stored in big-endian format, which is the most significant byte of a word in the low-address (leftmost) byte position. These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Step 4: Process message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; this module is labelled **F.**

- Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, H_{i-1}.
- Each round makes use of a 64-bit value, derived from the current 1024-bit block being processed *(Mi)*. These values are derived using a message schedule described subsequently. Each round also makes use of an additive constant *Ki*, where indicates one of the 80 rounds.
- These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers. The constants provide a "randomized" set of 64-bit patterns, which should eliminate any regularities in the input data.

- The output of the eightieth round is added to the input to the first round to produce (H_{i-1} to produce H_i. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in H_{i-1} using addition modulo 2^{64}.

Step 5: Output. After all 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.

Now let use summarize the behaviour of SHA-512 as follows:

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i)$$
$$MD = H_N$$

where

IV	=	initial value of the abcdefgh buffer, defined in step 3
abcdefgh_i	=	the output of the last round of processing of the ith message block
N	=	the number of blocks in the message (including padding and length fields)
SUM_{64}	=	addition modulo 2^{64} performed separately on each word of the pair of inputs
MD	=	final message digest value

MESSAGE AUTHENTICATION CODES & HASH FUNCTIONS

Message Authentication Requirements

1. Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key

2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

3. Masquerade: Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.

4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6. Timing modification: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., data-gram) could be delayed or replayed.

7. Source repudiation: Denial of transmission of message by source.

8. Destination repudiation: Denial of receipt of message by destination.

Message Authentication Code (MAC)

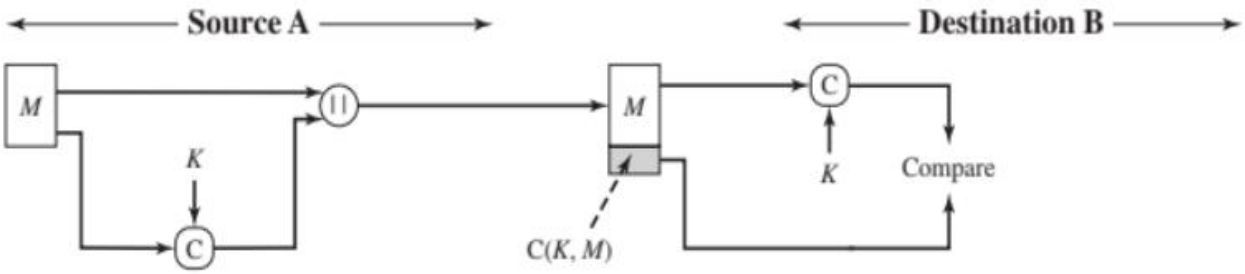
- **Message authentication** is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.
- A **message authentication code** (MAC) is an algorithm that requires the use of a secret key. A MAC takes a variable-length message and a secret key as input and produces an authentication code. A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.

This technique assumes that two communicating parties, say A and B, share a common secret key. When A has a message to send to B, it calculates the MAC as a function of the message and the key: ***MAC = MAC (K, M)***

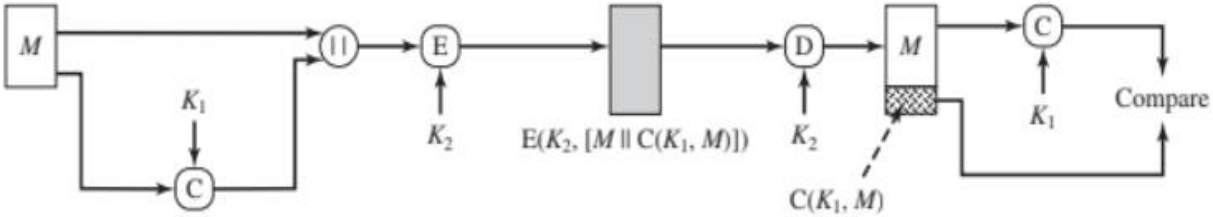
where,

M= input message
C = MAC function
K= shared secret key
MAC = message authentication code

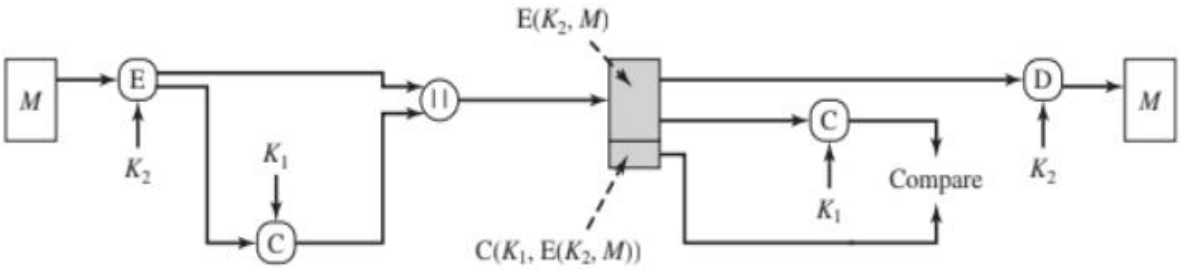
- The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.
- The received MAC is compared to the calculated MAC. If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then the receiver is assured that the message has not been altered.
- If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Digital Signatures and Authentication

- 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.

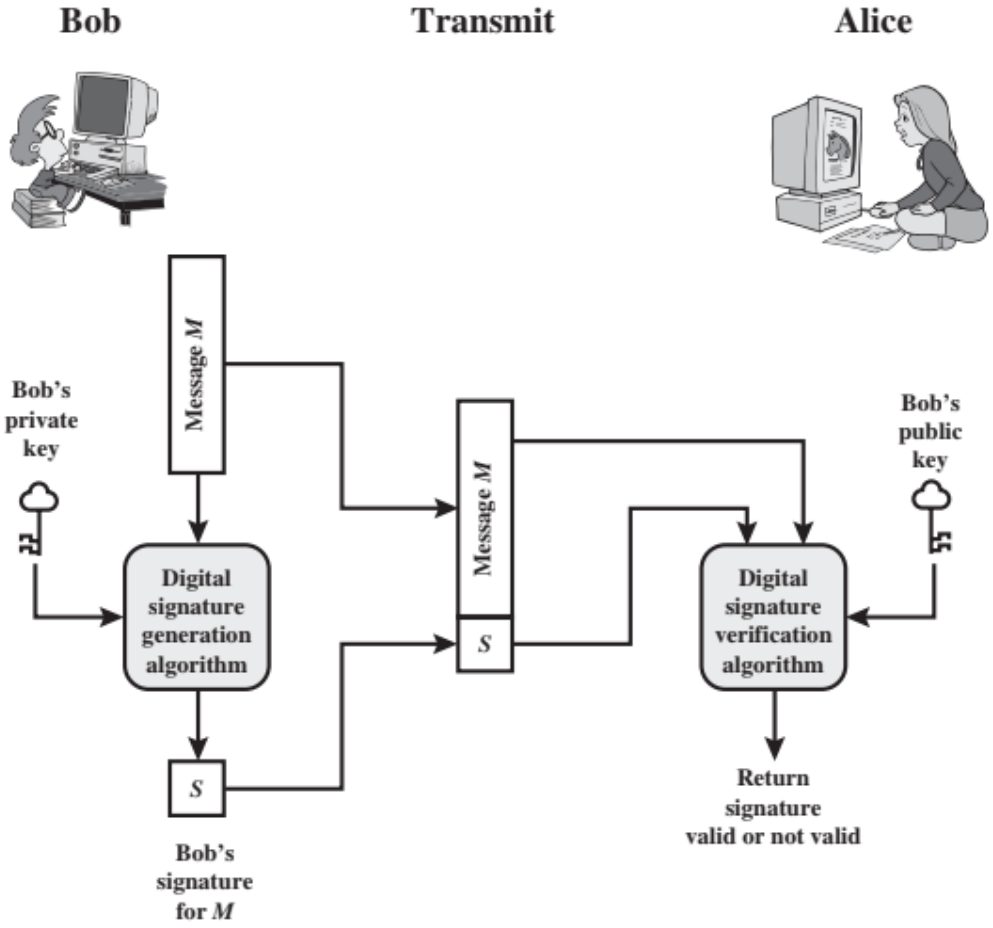


Fig: Generic Model of Digital Signature Process

For example, suppose that John sends an authenticated message to Mary, using one of the schemes. Consider the following disputes that could arise.

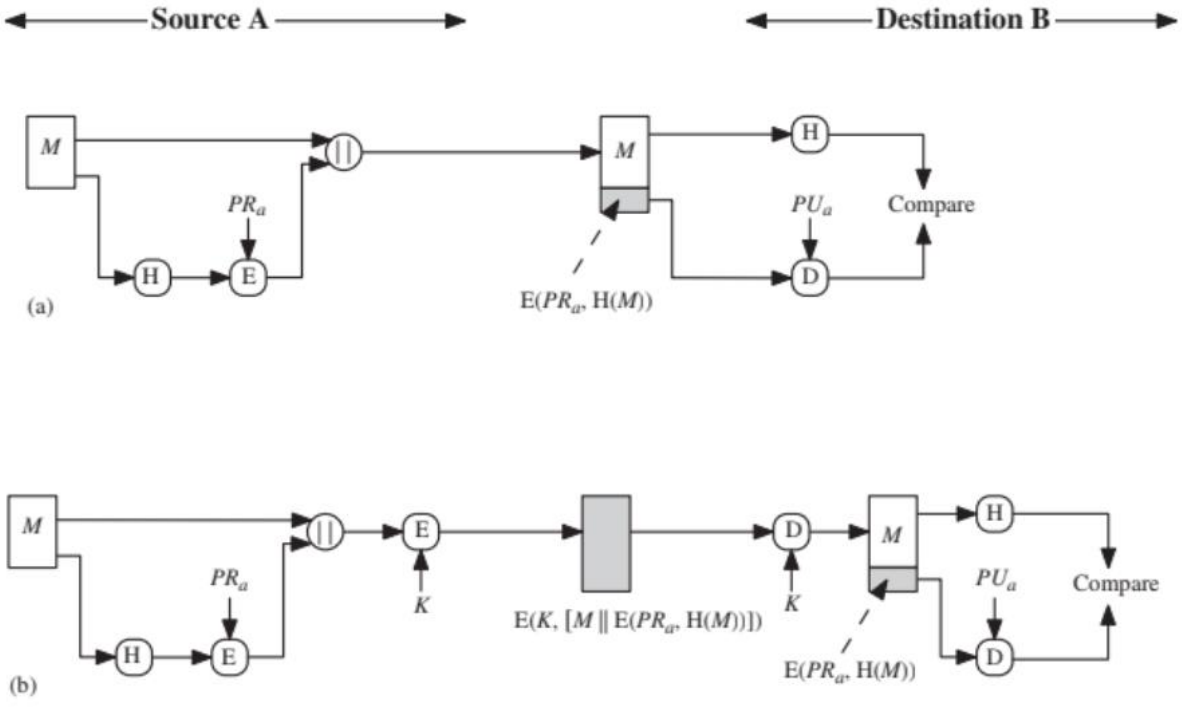
1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

Both scenarios are of legitimate concern.

- Here is an example of the first scenario: An electronic funds transfer takes place, and the receiver increases the amount of funds transferred and claims that the larger amount had arrived from the sender.
- An example of the second scenario is that an electronic mail message contains instructions to a stockbroker for a transaction that subsequently turns out badly. The sender pretends that the message was never sent.
- In situations where there is not 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:
  - It must verify the author and the date and time of the signature.
  - It must authenticate the contents at the time of the signature.
  - It must be verifiable by third parties, to resolve disputes.
  - Thus, the digital signature function includes the authentication function.

The operation of the digital signature is similar to that of the 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. In this case, an attacker who wishes to alter the message would need to know the user's private key.



Above figure illustrates, in a simplified fashion, how a hash code is used to provide a digital signature.

a. The hash code is encrypted, using public-key encryption with the sender's private key. As with figure above, this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

b. If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.

Digital Signature Standard (DSS)

- The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the **Digital Signature Standard (DSS).** The DSS makes use of the Secure Hash Algorithm (SHA) and presents a new digital signature technique, the Digital Signature Algorithm (DSA).
- The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

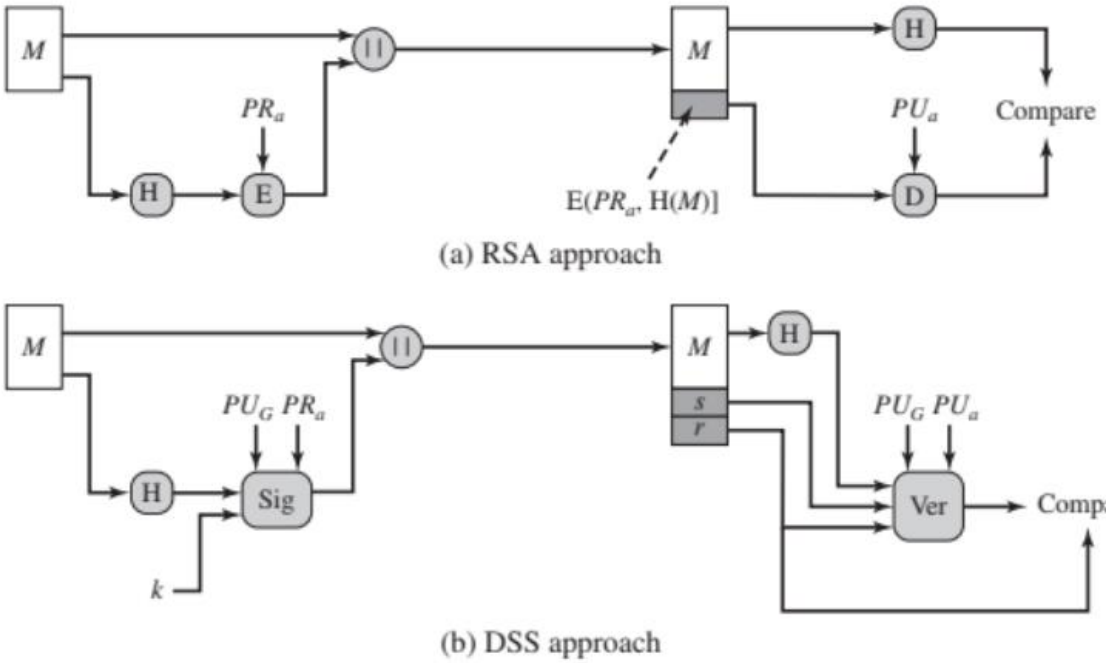


Figure 13.3 Two Approaches to Digital Signatures

- The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange.
- Above given Figure contrasts the DSS approach for generating digital signatures to that used with RSA.
- In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code

matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

- The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number generated for this particular signature. The signature function also depends on the sender's private key and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key. The result is a signature consisting of two components, labelled *s and r.*

Authentication Applications

A Key Distribution Scenario

The scenario assumes that each user shares a unique master key with the key distribution centre (KDC).

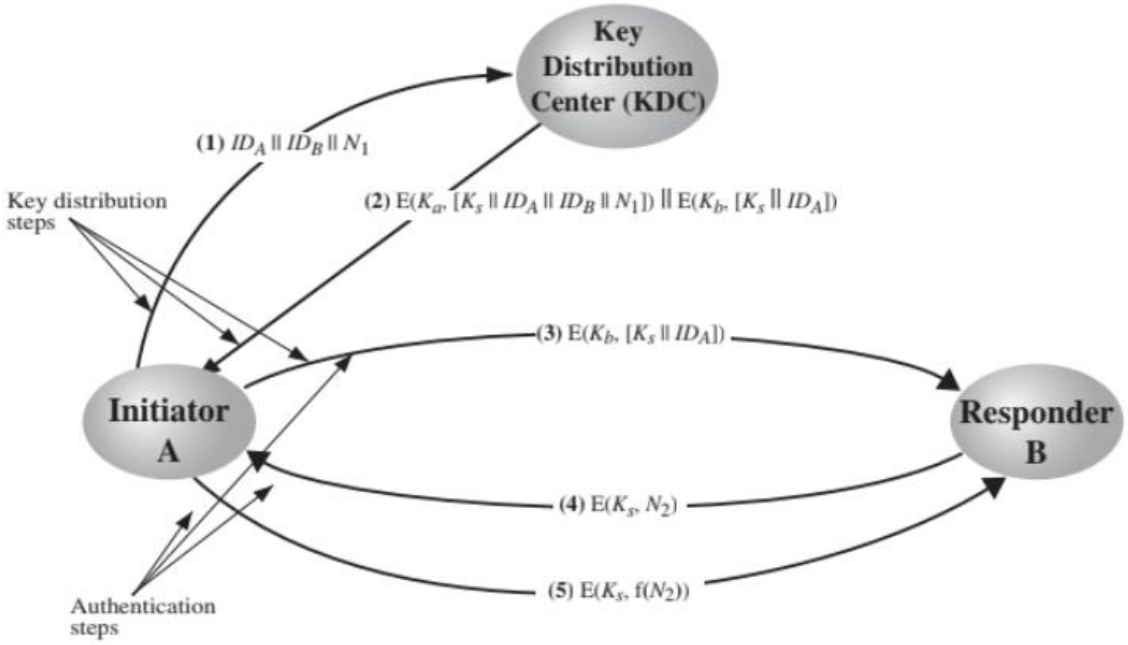


Fig: Key Distribution Scenario

Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key, K_a, known only to itself and the **KDC**; similarly, B shares the master key K_b with the KDC. The following steps occur.

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1, for this transaction, which we refer to as a **nonce**. The nonce may be a time-stamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. The KDC responds with a message encrypted using K_a. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

- The one-time session key, K_s, to be used for the session
- The original request message, including the nonce, to enable A to match this response with the appropriate request

Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request. In addition, the message includes two items intended for B:

- The one-time session key, K_s, to be used for the session
- An identifier of A (e.g., its network address), ID_A

These last two items are encrypted with K_b (the master key that the KDC shares with B).They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely $\mathbf{E(K_b, [K_s||ID_A])}$ Because this information is encrypted with k_b, it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).

At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N_2 to A.

5. Also, using K_s A responds with $f(N_2)$, where f is a function that performs some transformation on N_2 (e.g., adding one). These steps assure B that the original message it received (step 3) was

not a replay. Note that the actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function.

Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure

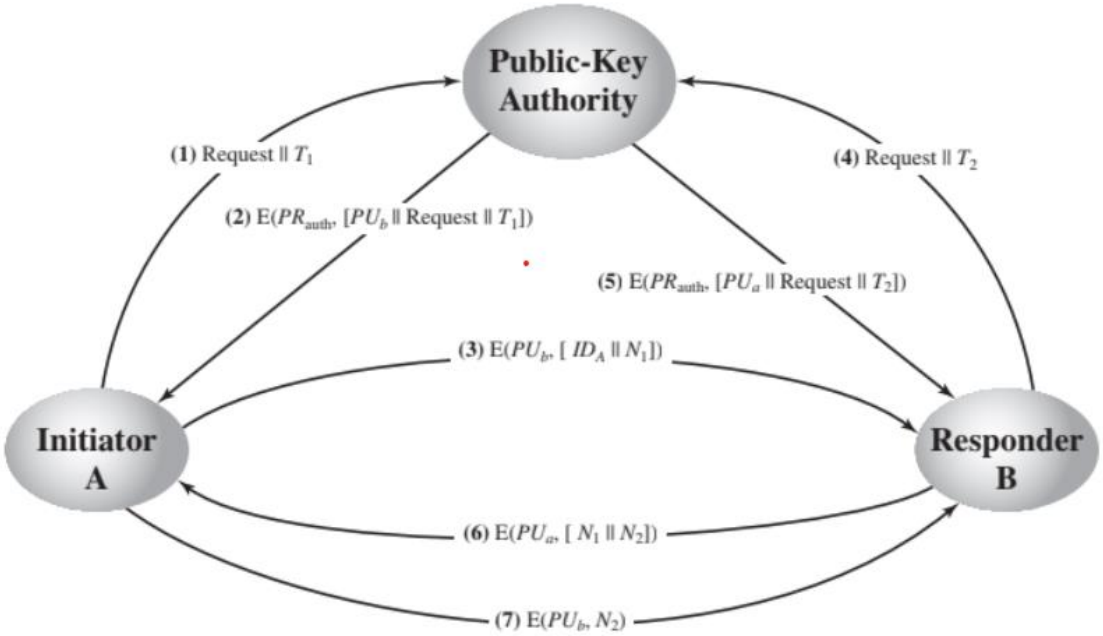


Fig: Public-Key Distribution Scenario

1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key PR_{auth}. Thus, A is able to decrypt the message using the authority's public key. Therefore A is assured that the message originated with the authority. The message includes the following:

- B's public key, PU_b, which A can use to encrypt messages destined for B
- The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority.
- The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A *(ID_A)* and a nonce *(N_1),* which is used to identify this transaction uniquely.
4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:
5. B sends a message to A encrypted with PU_a and containing A's nonce *(N_1),* as well as a new nonce generated by B *(N_2)*. Because only B could have decrypted message (3), the presence of *(N_1),* in message (6) assures A that the correspondent is B.
7. A returns *(N_2),* which is encrypted using B's public key, to assure B that its correspondent is A.

Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use—a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

X.509 Certificates

- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates.
- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.
- X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA.
- The digital signature scheme is assumed to require the use of a hash function. Again, the standard does not dictate a specific hash algorithm.
- The X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts.
- The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.
- The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

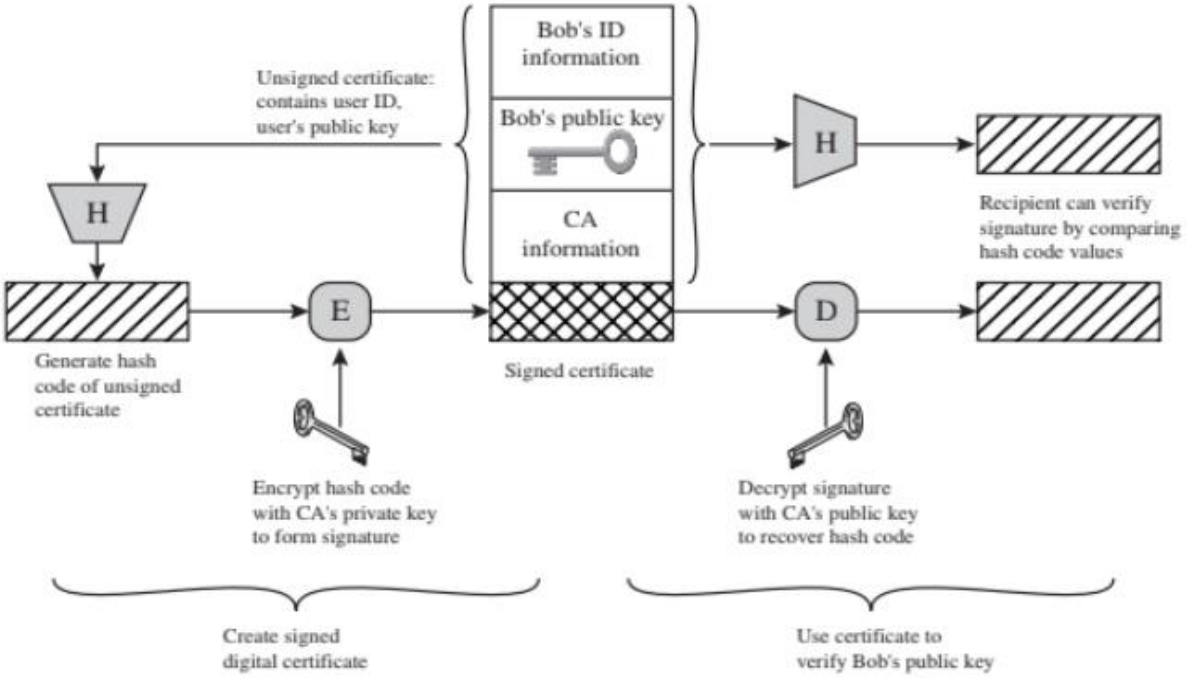
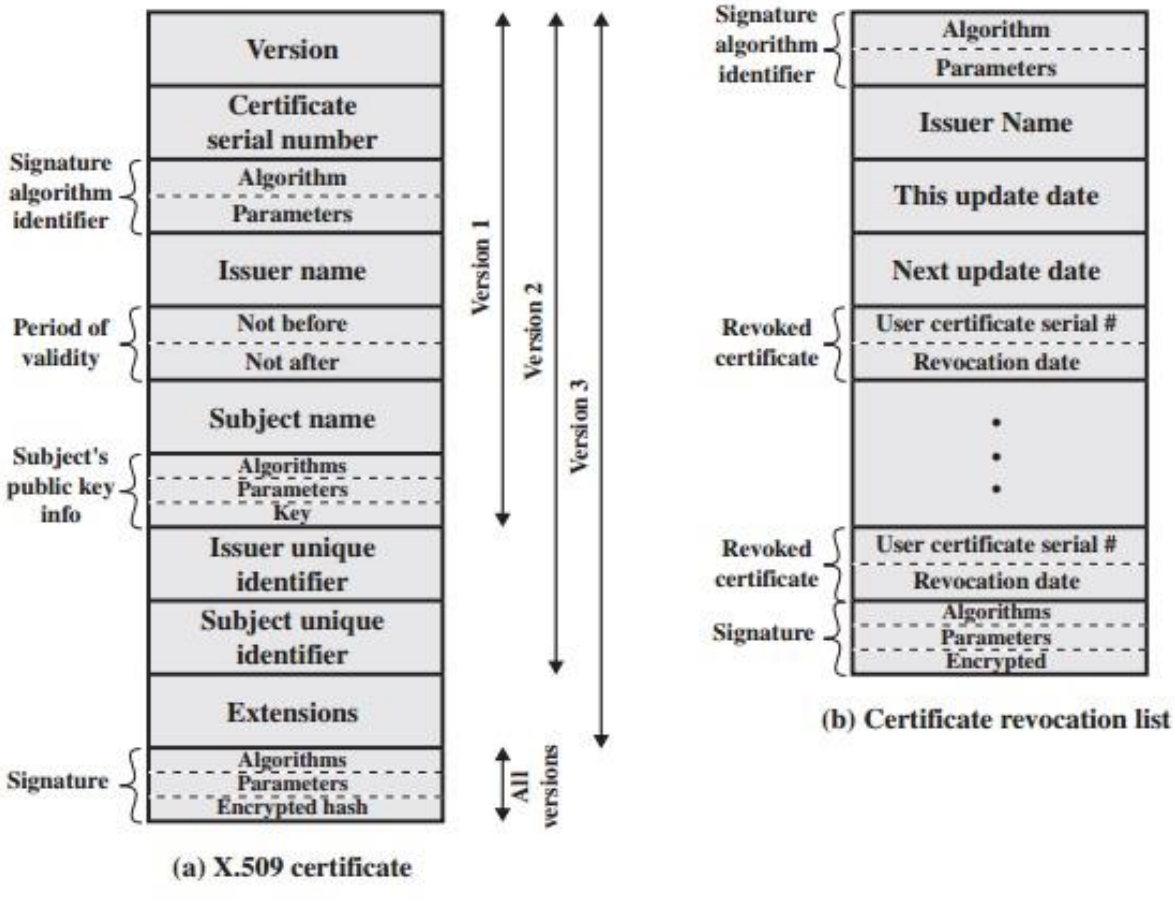


Fig: Public-Key Certificate Use



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

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

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

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

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

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

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

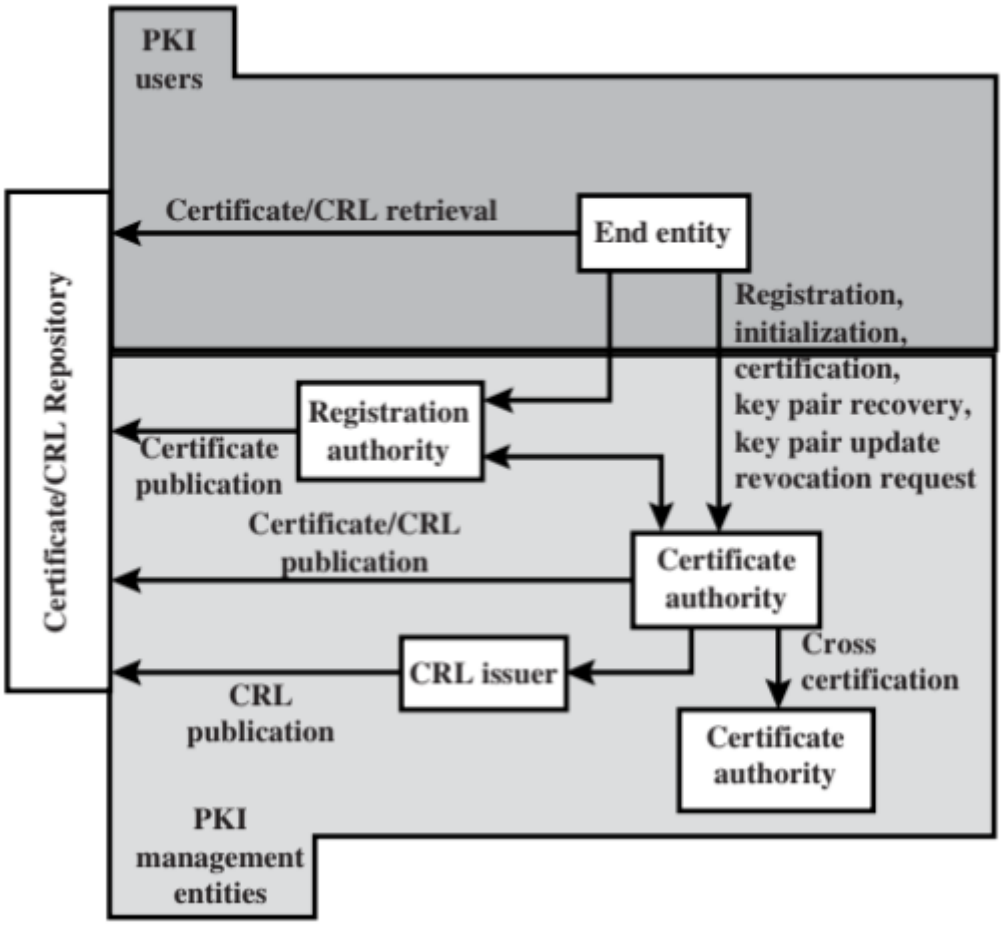
Issuer unique identifier: An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

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

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

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

Public-Key Infrastructure



- RFC 2822 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.
- The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys.
- X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet.
- This section describes the PKIX model; The key elements of the PKIX model. These elements are:

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

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

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

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

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