SCS1316 - NETWORK SECURITY

UNIT - II CRYPTOGRAPHY BASICS

Course Objectives:

- To understand different types of vulnerabilities, threats and attacks.
- To get familiarize in basic number theory and cryptography related to network security issues.

Course Outcome:

- Encrypt and decrypt using basic cryptographic algorithms

Syllabus:

Terminologies – Cryptography – Classification: based on operation, number of keys used, Processing - Crypt analysis: Types - Classical Encryption - Substitution Cipher: Ceaser Cipher, Brute Force attack, Vignere Cipher, One time pad, Transposition Cipher: Rail fence Cipher, Simple row column Transfer, Play Fair Cipher, 2X2 Hill cipher - Stream cipher - Block Cipher - Modes of operation – DES – AES - RSA algorithm

Terminologies of Cryptography:

Cryptography

The many schemes used for encryption constitute the area of study known as cryptography

Crypt analysis

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.”

Cryptology

The areas of cryptography and cryptanalysis together are called cryptology

Cipher

Encryption scheme is known as a cryptographic system or a cipher

Plain Text

This is the original intelligible message or data that is fed into the algorithm as input.

Cipher Text

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

cipher texts. The cipher text is an apparently random stream of data and, as it stands, is unintelligible.

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.

Encryption

The process of converting from plaintext to cipher text

Decryption

The process of restoring the plaintext from the cipher text

Enciphering Algorithm

The encryption algorithm performs various substitutions and transformations on the plaintext

Deciphering Algorithm

This is essentially the encryption algorithm run in reverse. It takes the cipher text and the secret key and produces the original plaintext.

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 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.

Principles of Security

Symmetric Cipher Model

A symmetric encryption scheme has five ingredients. They are Plain Text, Encryption Algorithm, Secret Key, Decryption Algorithm, Cipher Text

There are two requirements for secure use of conventional encryption:

- We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more cipher texts would be unable to decipher the cipher text or figure out the key.
- Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

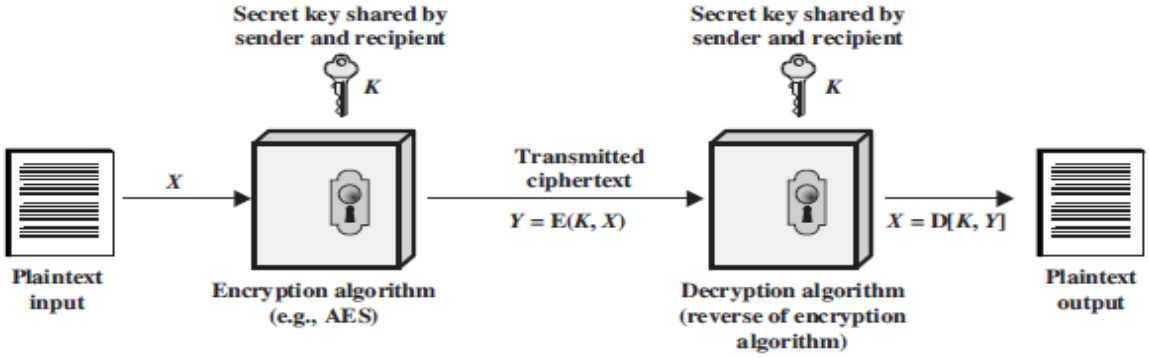


Fig. 1.1a Model of Symmetric Encryption

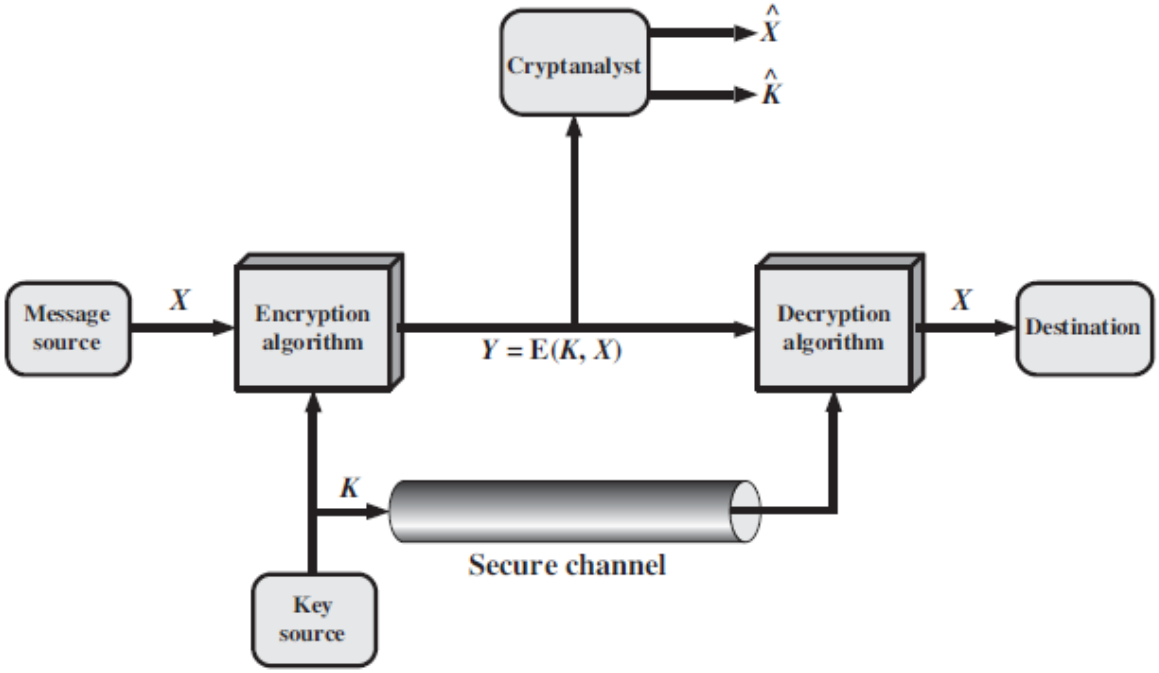


Fig. 1.1b Model of Symmetric Cryptosystem

Confidentiality: This term covers two related concepts:

- Data 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.

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.

Integrity: 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.

Non-repudiation

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

- Nonrepudiation, Origin: Proof that the message was sent by the specified party
- Nonrepudiation, Destination: Proof that the message was received by the specified party

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

Availability

Assures that systems work promptly and service is not denied to authorized users.

Classical Encryption Techniques

Substitution Cipher

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 cipher text bit patterns.

Caesar Cipher

The earliest known use of a substitution cipher, and the simplest, was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.

For example,

Plain text : meet me after the toga party
Cipher Text : PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

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

Let us assign a numerical equivalent to each letter:

a	**b**	**c**	**d**	**e**	**F**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
0	1	2	3	4	5	6	7	8	9	10	11	12
n	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows.
For each plaintext letter p, substitute the cipher text letter C
C = E(3, p) = (p + 3) mod 26

A shift may be of any amount, so that the general Caesar algorithm is
C = E(k, p) = (p + k) mod 26
where k takes on a value in the range 1 to 25.

The decryption algorithm is simply
p = D(k, C) = (C k) mod 26
If it is known that a given cipher text is a Caesar cipher, then a brute-force cryptanalysis is easily performed: Simply try all the 25 possible keys.

Three important characteristics of this problem enabled us to use a brute-force cryptanalysis:

- The encryption and decryption algorithms are known.
- There are only 25 keys to try.
- The language of the plaintext is known and easily recognizable.

Play Fair Cipher

The best-known multiple-letter encryption cipher is the Play fair, which treats **digrams** in the plaintext as single units and translates these units into cipher text digrams. The Play fair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword.

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

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:

- Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that ba**ll**oon would be treated as ba **lx l**o on
- 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**
- 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**
- 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)

Hill cipher

Another interesting multi-letter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. The encryption algorithm takes m successive plaintext letters and substitutes for them m cipher text letters.

The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1 ... z = 25).
For m = 3, the system can be described as follows:

$c_1 = (k_{11}P_1 + k_{12}P_2 + k_{13}P_3) \bmod 26$
$c_2 = (k_{21}P_1 + k_{22}P_2 + k_{23}P_3) \bmod 26$
$c_3 = (k_{31}P_1 + k_{32}P_2 + k_{33}P_3) \bmod 26$

This can be expressed in term of column vectors and matrices:

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} P_1 & P_2 & P_3 \end{pmatrix} \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \text{mod} 26$$

Or

C = PK mod 26

where C and P are column vectors of length 3, representing the plaintext and cipher text, and K is a 3 x 3 matrix, representing the encryption key. Operations are performed in mod 26.

$P = D(K, C) = CK^{-1} \bmod 26 = PKK^{-1} = P$

One Time Pad

An Army Signal Corp officer, Joseph Mauborgne suggested using a **random key** that is as **long as the message**, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a one-time pad, is **unbreakable**.

It produces random output that bears no statistical relationship to the plaintext. Because the cipher text contains no information whatsoever about the plaintext, there is simply no way to break the code.

An example should illustrate our point. Suppose that we are using a 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the message.

Consider the
cipher text : ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
We now show two different decryptions using two different keys:

key 1: pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
plain text: mr mustard with the candlestick in the hall

key 2: mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt
plain text: miss scarlet with the knife in the library

If the actual key were produced in a truly random fashion, then the cryptanalyst cannot say that one of these two keys is more likely than the other. Thus, there is no way to decide which key is correct and therefore which plaintext is correct. Therefore, the code is unbreakable.

Transposition Cipher

A 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 Technique

The simplest transposition 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:

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 **“MEMATRHTGPRYETEFETEOAAT”**

Simple Columnar Technique

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

	1	2	3	4	5	6	7
1	*a*	*t*	*t*	*a*	*c*	*k*	*p*
2	*o*	*s*	*t*	*p*	*o*	*n*	*e*
3	*d*	*u*	*n*	*t*	*i*	*l*	*t*
4	*w*	*o*	*a*	*m*	*x*	*y*	*z*

Cipher text: **TTNAAPTMTSUOAODWCOIXKNLYPETZ**

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext.

For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the cipher text in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful.

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed.
Thus, if the foregoing message is re-encrypted using the same algorithm,

Key: 3 4 2 1 5 6 7

	1	2	3	4	5	6	7
1	*t*	*t*	*n*	*a*	*p*	*t*	*m*
2	*t*	*t*	*s*	*u*	*o*	*a*	*o*
3	*d*	*w*	*c*	*o*	*i*	*x*	*k*
4	*n*	*l*	*y*	*p*	*e*	*t*	*z*

Cipher text: **NSCYAUOPTTWLTTDNPOIETAXTMOKZ**

Comparison of Stream Ciphers and Block Ciphers

A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.

In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the keystream is as long as the plaintext bit stream. If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream. However, the keystream must be provided to both users in advance via some independent and secure channel. This introduces insurmountable logistical problems if the intended data traffic is very large.
Accordingly, for practical reasons, the bit-stream generator must be implemented as an algorithmic procedure, so that the cryptographic bit stream can be produced by both users. In this approach, the bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong. Now, the two users need only share the generating key, and each can produce the keystream.

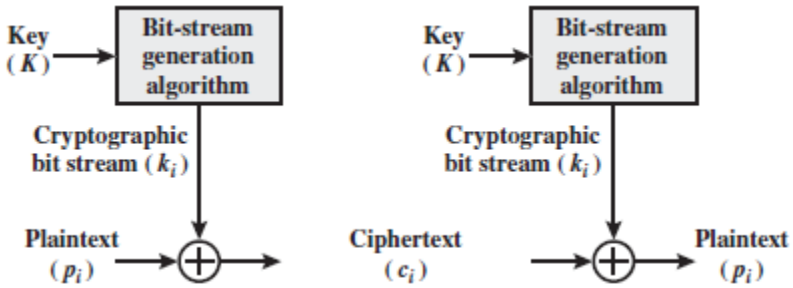


Fig.2.1 Stream Cipher using algorithmic bit-stream generator

A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically, a block size of 64 or 128 bits is used. In general, they seem applicable to a broader range of applications than stream ciphers. The vast majority of network-based symmetric cryptographic applications make use of block ciphers.

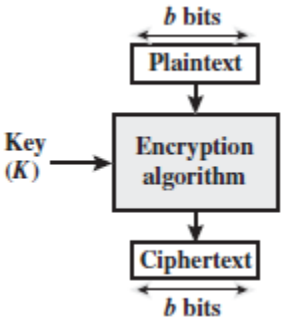


Fig.2.2 Block Cipher

Feistel Block 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

987possible 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.

In fact, Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions.

Diffusion: A cryptographic technique that seeks to obscure the statistical structure of the plaintext by spreading out the influence of each individual plaintext digit over many cipher text digits.

Confusion: A cryptographic technique that seeks to make the relationship between the statistics of the cipher text and the value of the encryption key as complex as possible. This is achieved by the use of a complex scrambling algorithm that depends on the key and the input.

Figure 2.3 depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length **2w** bits and a key .The plaintext block is divided into two halves L_0 and R_0. The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a subkey K_i derived from the overall K. In general, the subkeys K_i are different from K and from each other. In Figure 2.3, 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 K_i . Another way to express this is to say that F is a function of right-half block of w bits and a subkey of y bits, which produces an output value of length *w* bits: *F*(*REi*, *Ki*+1). Following this substitution, a

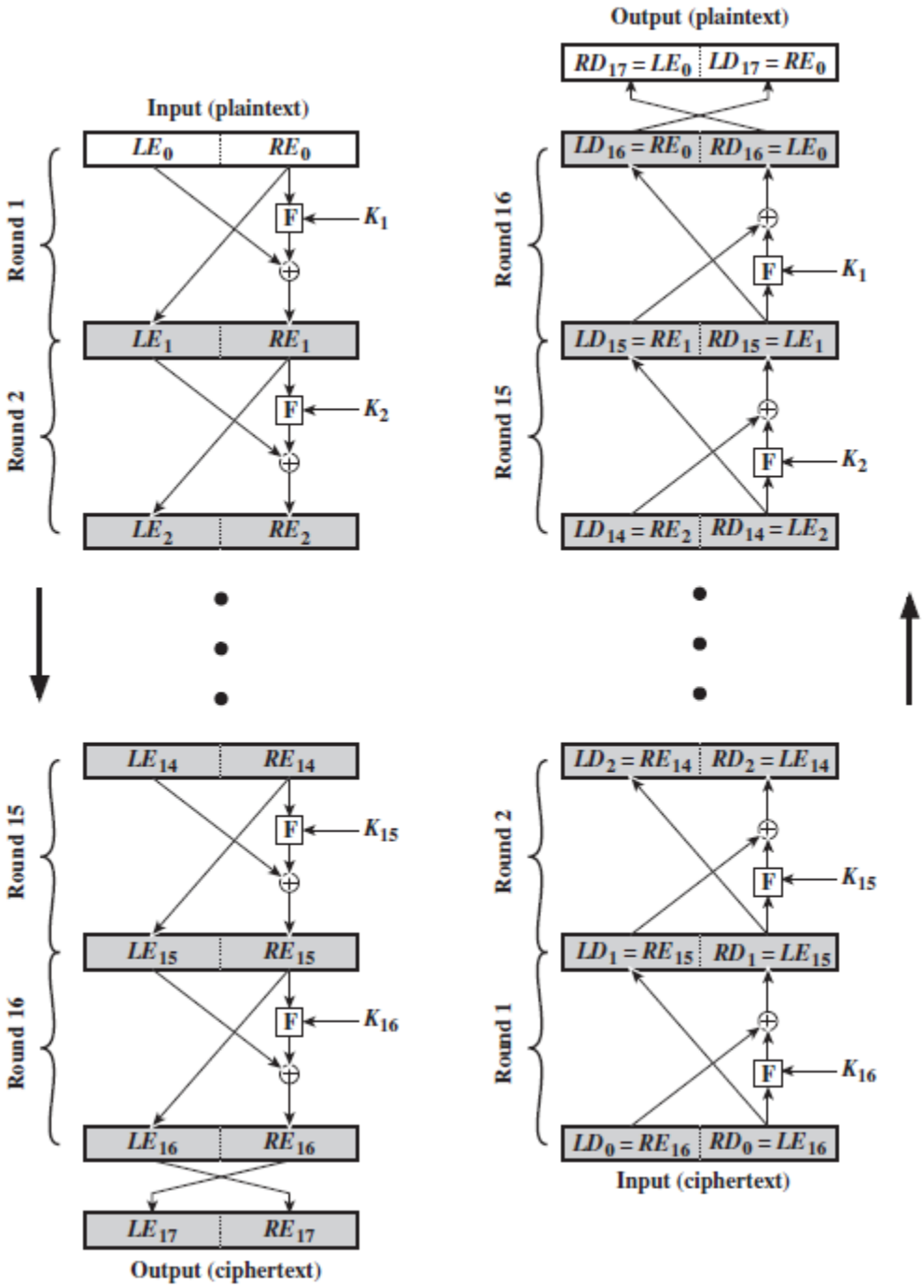


Fig.2.3 Feistel Encryption and Decryption

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.

There are two other considerations in the design of a Feistel cipher:
• **Fast software encryption/decryption:** In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
• **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality.

Block Cipher Modes of Operation

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. In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream.

(i) Electronic Code Book (ECB) Mode

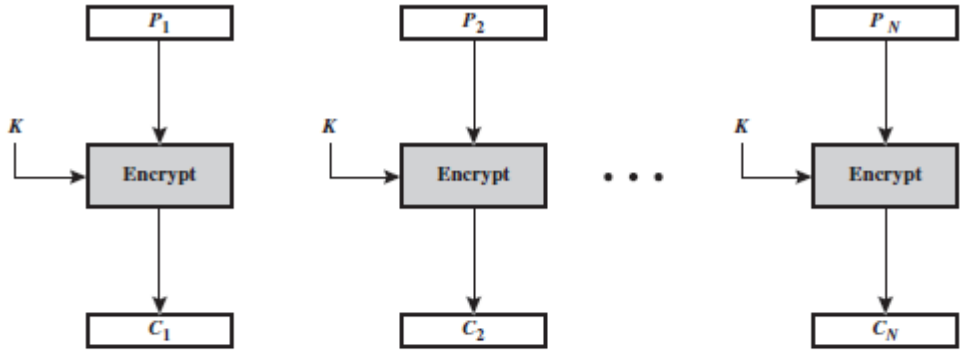
This mode is a most straightforward way of processing a series of sequentially listed message blocks.

Operation

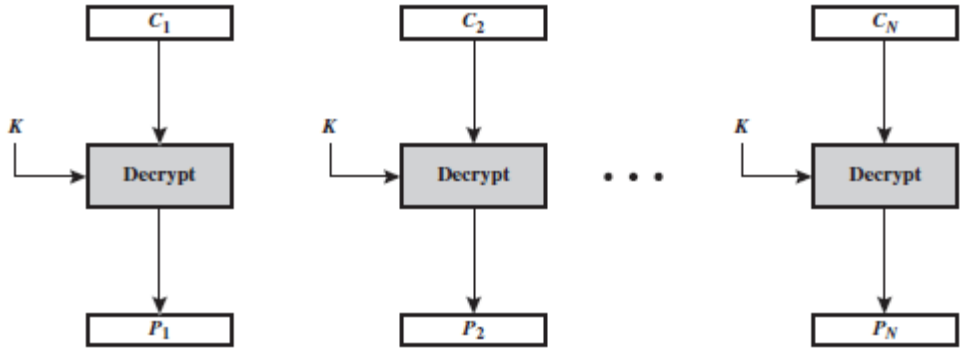
- The user takes the first block of plaintext and encrypts it with the key to produce the first block of cipher text.
- He then takes the second block of plaintext and follows the same process with same key and so on so forth.

The ECB mode is **deterministic**, that is, if plaintext block P1, P2,…, Pm are encrypted twice under the same key, the output cipher text blocks will be the same.
In fact, for a given key technically we can create a codebook of cipher texts for all possible plaintext blocks. Encryption would then entail only looking up for required plaintext and select the corresponding cipher text. Thus, the operation is analogous to the assignment of code words in a codebook, and hence gets an official name: Electronic Codebook mode of operation (ECB). It is illustrated as follows:

ECB	$C_j = E(K, P_j)$ $j = 1, \ldots, N$	$P_j = D(K, C_j)$ $j = 1, \ldots, N$



(a) Encryption



(b) Decryption

(ii) Cipher Block Chaining (CBC) Mode

CBC mode of operation provides message dependence for generating ciphertext and makes the system non-deterministic.

Operation

The operation of CBC mode is depicted in the following illustration. The steps are as follows:

- Load the n-bit Initialization Vector (IV) in the top register
- XOR the n-bit plaintext block with data value in top register
- Encrypt the result of XOR operation with underlying block cipher with key K.
- Feed cipher text block into top register and continue the operation till all plaintext blocks are processed
- For decryption, IV data is XORed with first cipher text block decrypted. The first cipher text block is also fed into to register replacing IV for decrypting next cipher text block

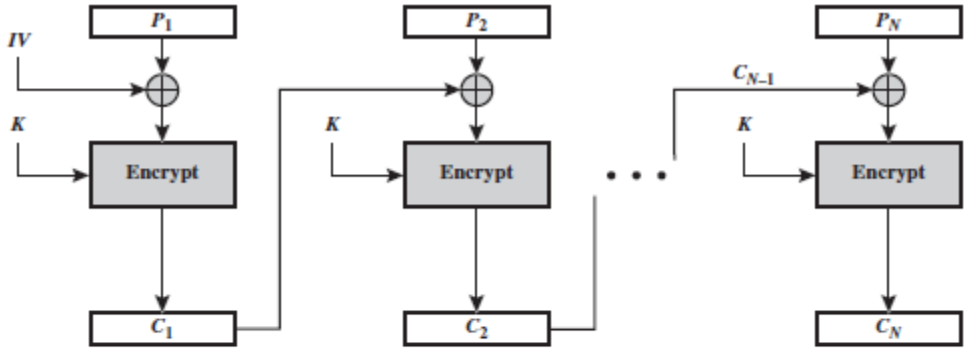
$$C_j = \mathrm{E}(K, [C_{j-1} \oplus P_j])$$

$$\mathrm{D}(K, C_j) = \mathrm{D}(K, E(K, [C_{j-1} \oplus P_j]))$$

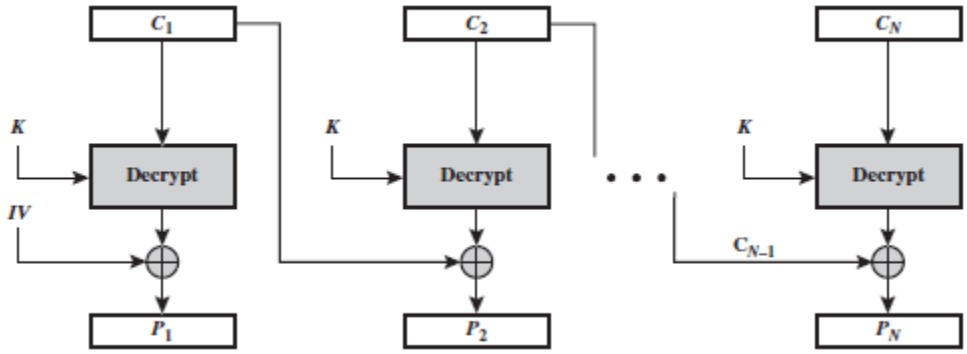
$$\mathrm{D}(K, C_j) = C_{j-1} \oplus P_j$$

$$C_{j-1} \oplus \mathrm{D}(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

CBC	$C_1 = \mathrm{E}(K, [P_1 \oplus \mathrm{IV}])$ $C_j = \mathrm{E}(K, [P_j \oplus C_{j-1}])\ \ j = 2, \ldots, N$	$P_1 = \mathrm{D}(K, C_1) \oplus \mathrm{IV}$ $P_j = \mathrm{D}(K, C_j) \oplus C_{j-1}\ \ \ j = 2, \ldots, N$



(a) Encryption



(b) Decryption

(iii) Cipher Feedback (CFB) Mode

In this mode, each ciphertext block gets 'fed back' into the encryption process in order to encrypt the next plaintext block.

Operation

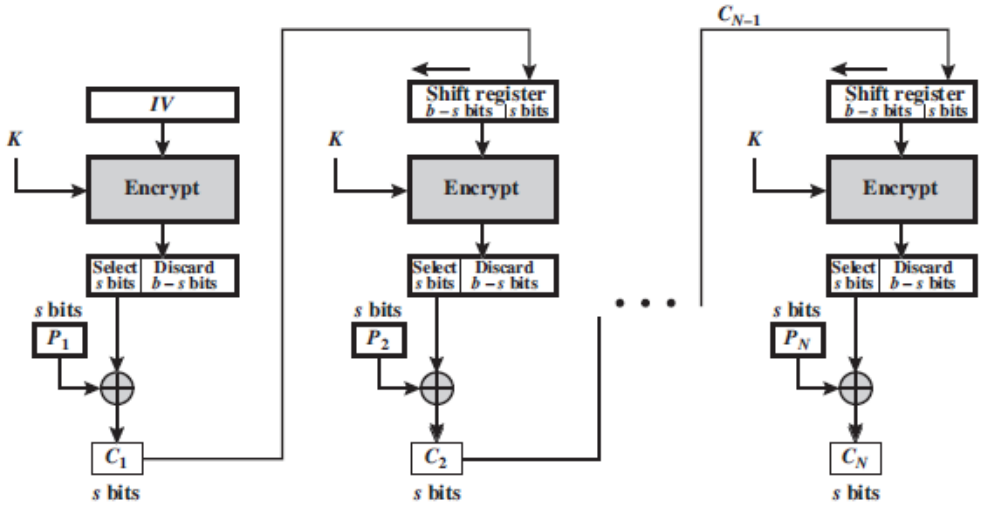
The operation of CFB mode is depicted in the following illustration. For example, in the present system, a message block has a size 's' bits where 1 < s < n. The CFB mode requires an initialization vector (IV) as the initial random n-bit input block. The IV need not be secret. Steps of operation are:

- Load the IV in the top register
- Encrypt the data value in top register with underlying block cipher with key K
- Take only 's' number of most significant bits (left bits) of output of encryption process and XOR them with 's' bit plaintext message block to generate cipher text block
- Feed cipher text block into top register by shifting already present data to the left and continue the operation till all plaintext blocks are processed
- Essentially, the previous cipher text block is encrypted with the key, and then the result is XORed to the current plaintext block
- Similar steps are followed for decryption. Pre-decided IV is initially loaded at the start of decryption

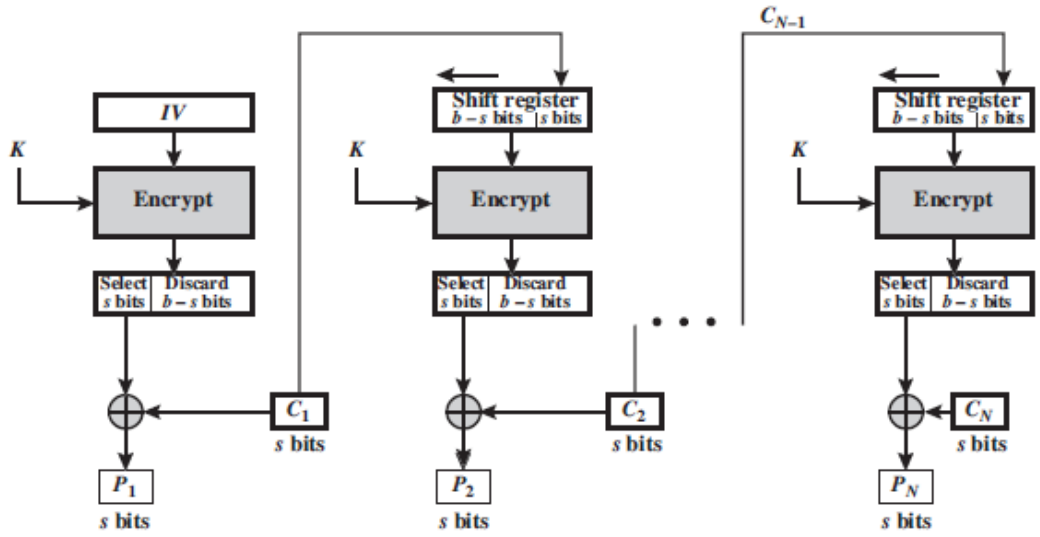
$$C_1 = P_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

$$P_1 = C_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

CFB	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1}$ $j = 2, \ldots, N$ $O_j = \text{E}(K, I_j)$ $j = 1, \ldots, N$ $C_j = P_j \oplus \text{MSB}_s(O_j)$ $j = 1, \ldots, N$	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1}$ $j = 2, \ldots, N$ $O_j = \text{E}(K, I_j)$ $j = 1, \ldots, N$ $P_j = C_j \oplus \text{MSB}_s(O_j)$ $j = 1, \ldots, N$



(a) Encryption



(b) Decryption

(iv)Output Feedback (OFB) Mode

It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key-stream generator as in case of CFB mode.

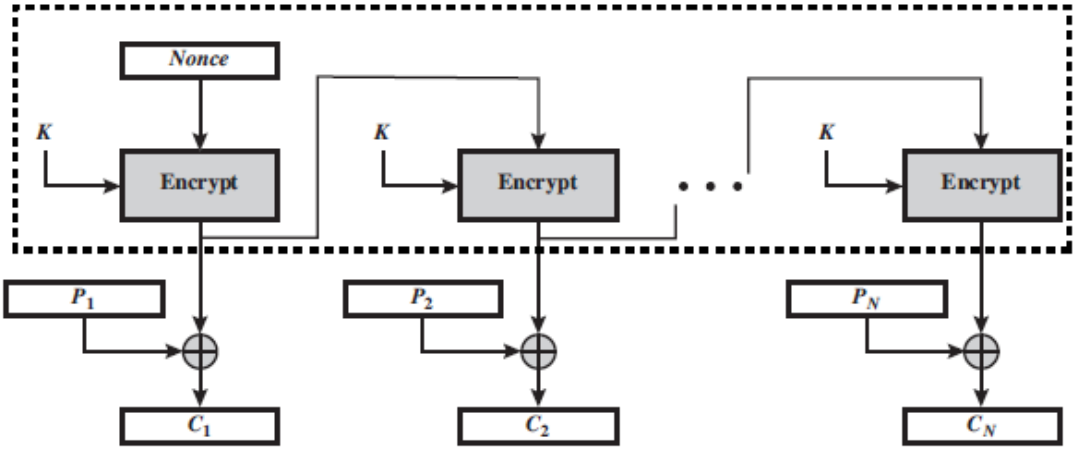
The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

The operation is depicted in the following illustration:

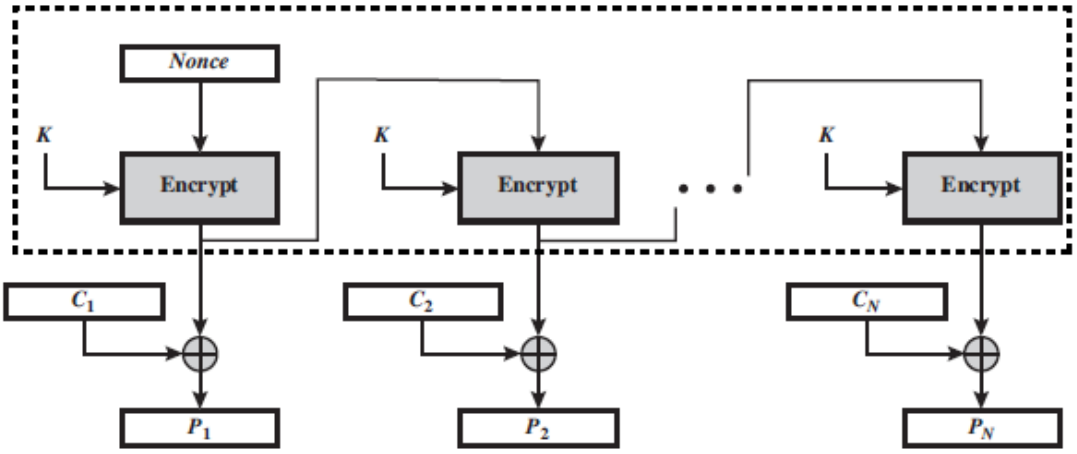
$$C_j = P_j \oplus \text{E}(K, [C_{j-i} \oplus P_{j-1}])$$

$$P_j = C_j \oplus \text{E}(K, [C_{j-1} \oplus P_{j-1}])$$

OFB	$I_1 = Nonce$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u(O_N)$	$I_1 = Nonce$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u(O_N)$



(a) Encryption



(b) Decryption

(v) Counter (CTR) Mode

It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a cipher text block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

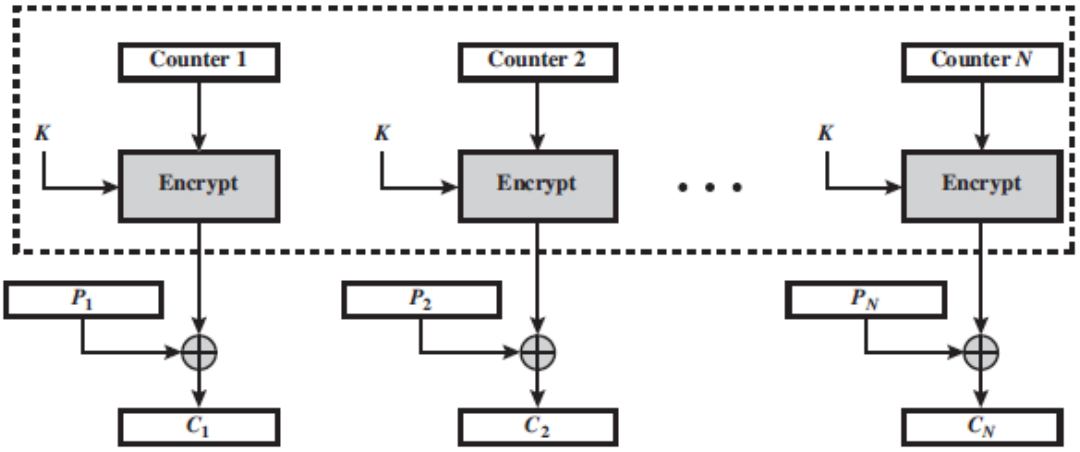
Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are:

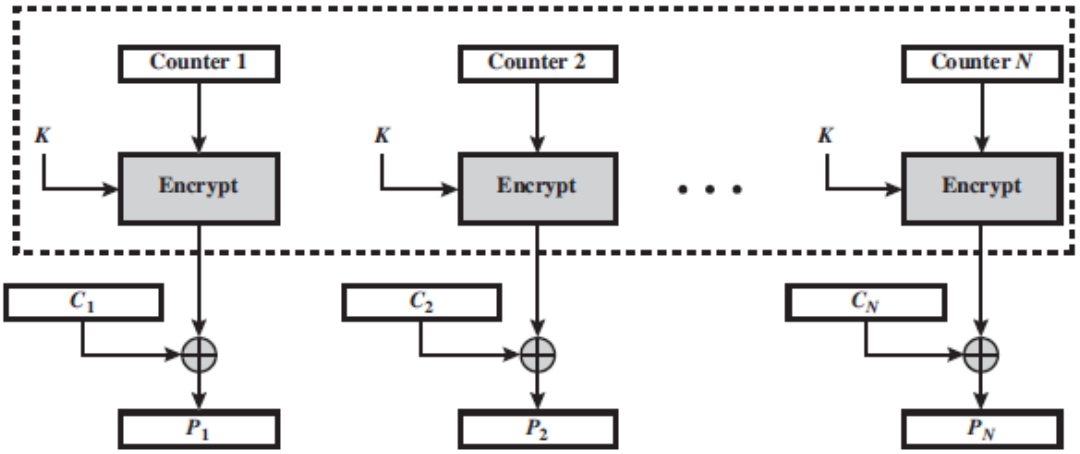
- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode
- Encrypt the contents of the counter with the key and place the result in the bottom register
- Take the first plaintext block P1 and XOR this to the contents of the bottom register

- The result of this is C1. Send C1 to the receiver and update the counter. The counter
- update replaces the cipher text feedback in CFB mode
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The cipher text block is XORed with the output of encrypted contents of counter value. After decryption of each cipher text block counter is updated as in case of encryption

CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$



(a) Encryption



(b) Decryption

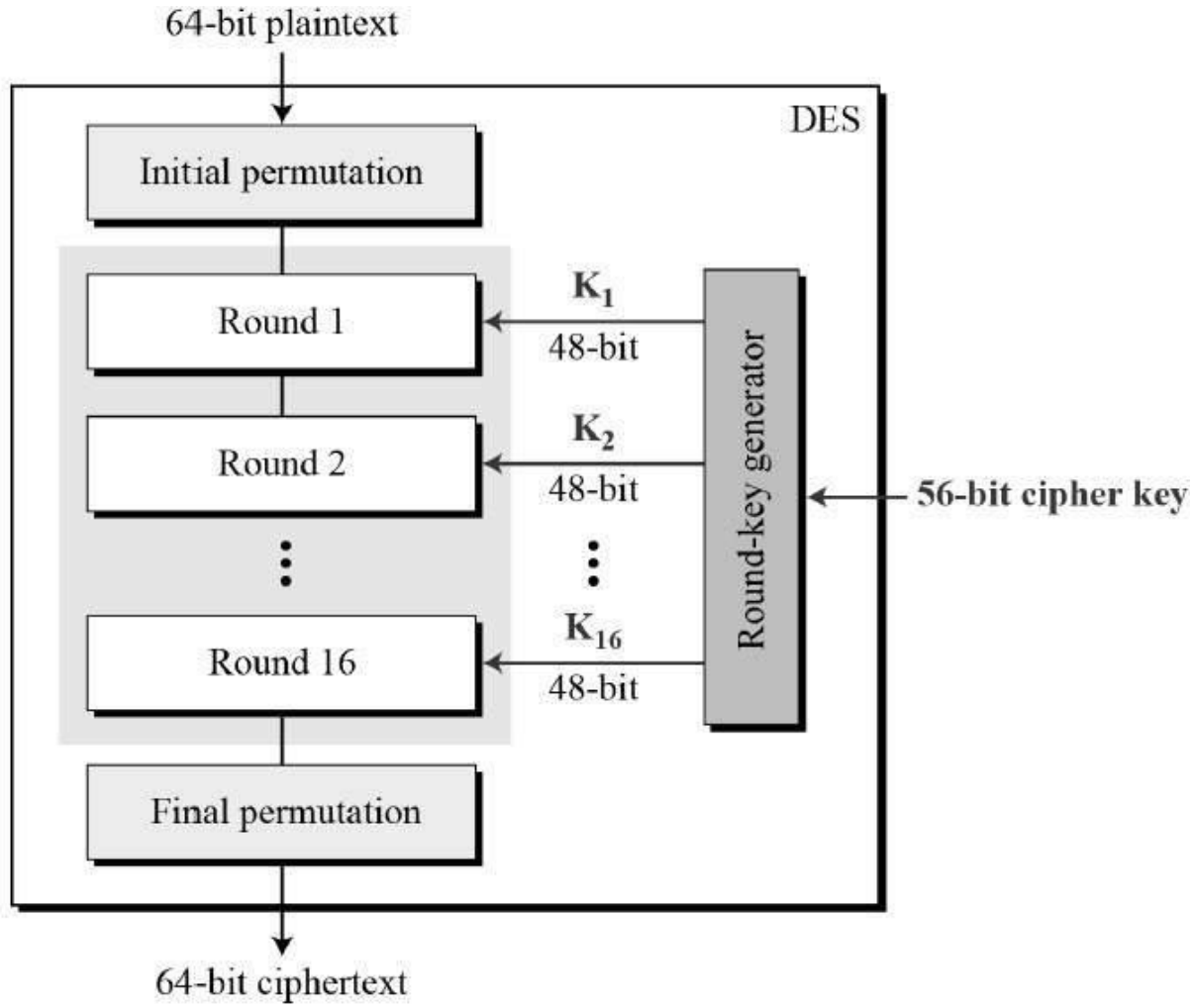
Table: Block Cipher Modes of Operation

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

Data Encryption Standard

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration :



Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

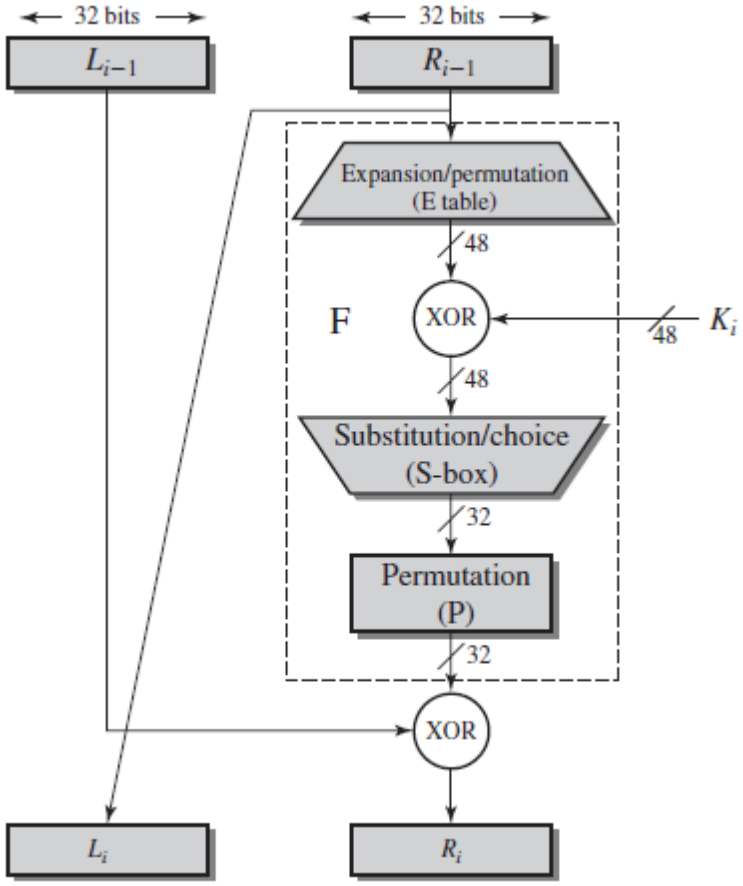
Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −

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

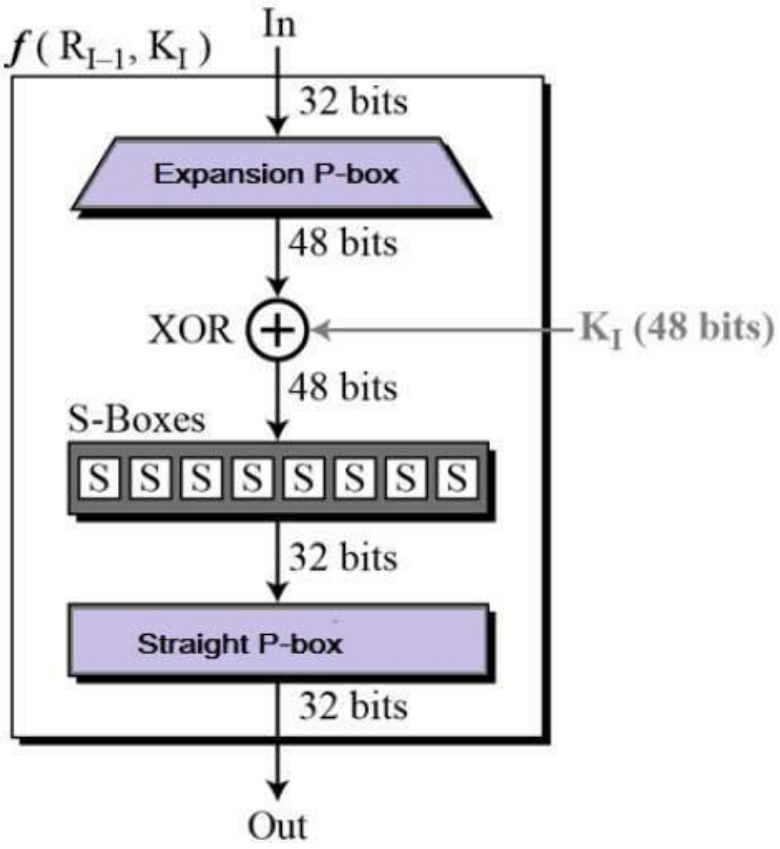
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Details of one round in DES

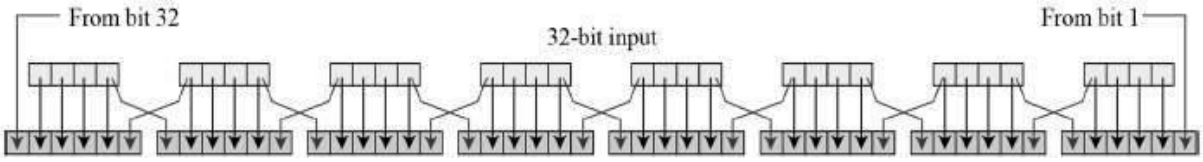


Round Function (F)

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



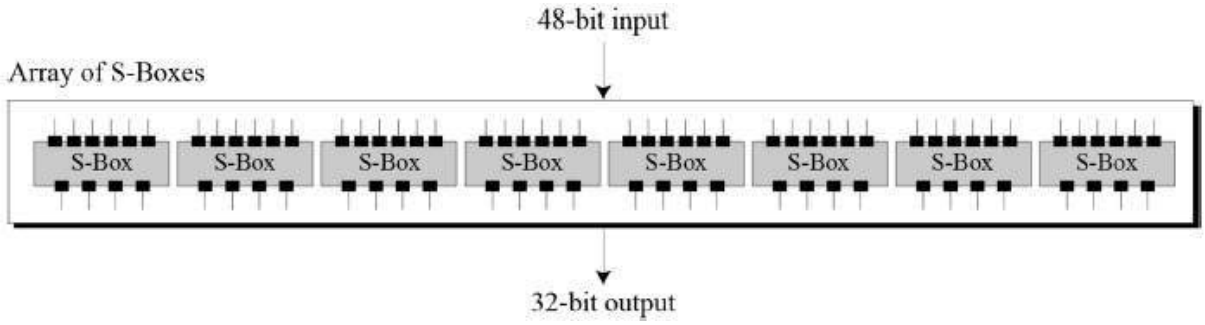
- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration



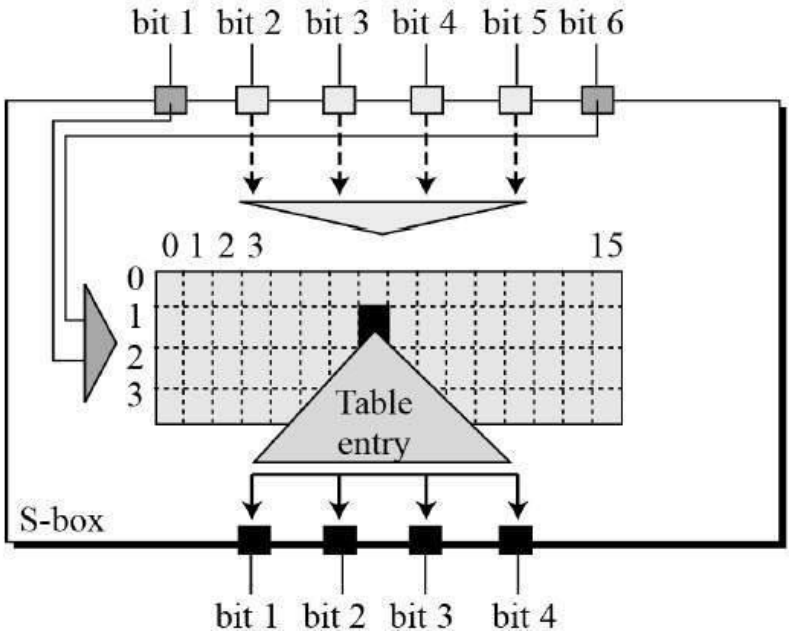
- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown

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

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration



The S-box rule is illustrated below



- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

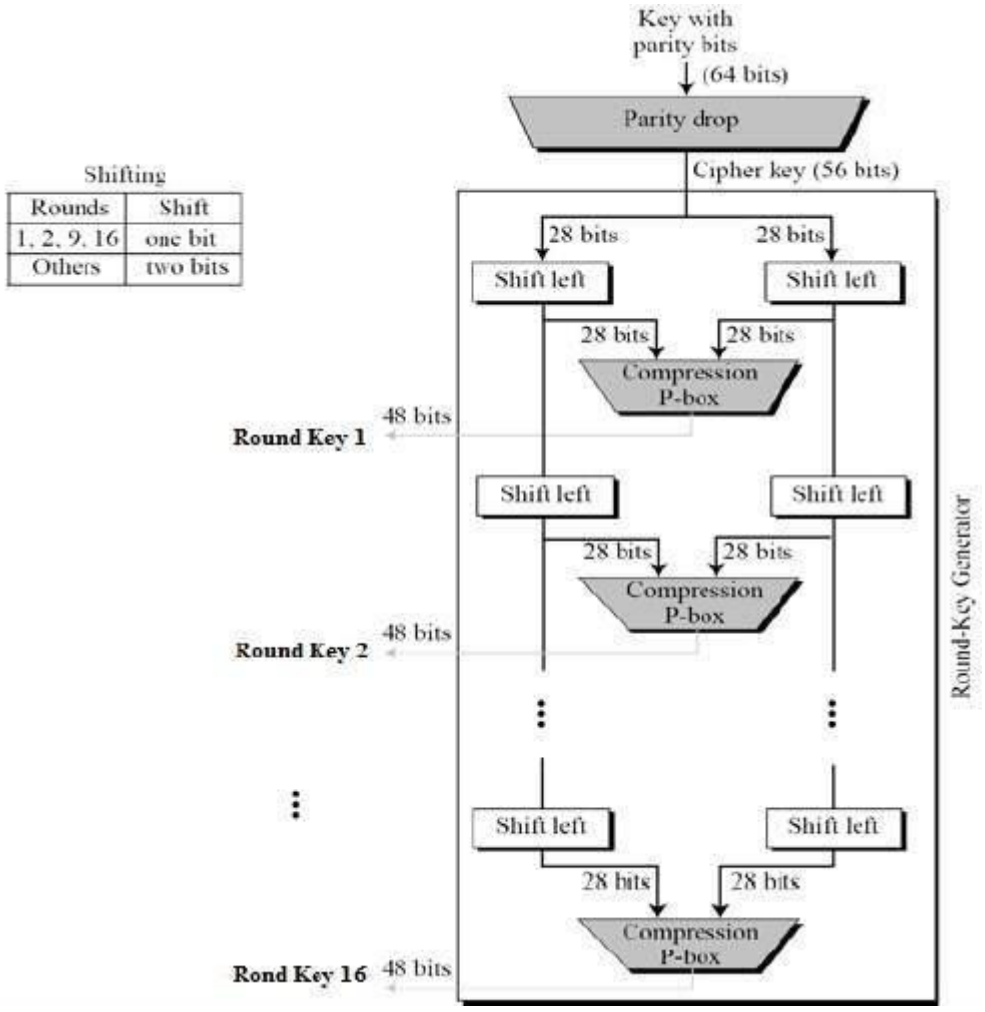
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

- **Straight Permutation** − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

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

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration



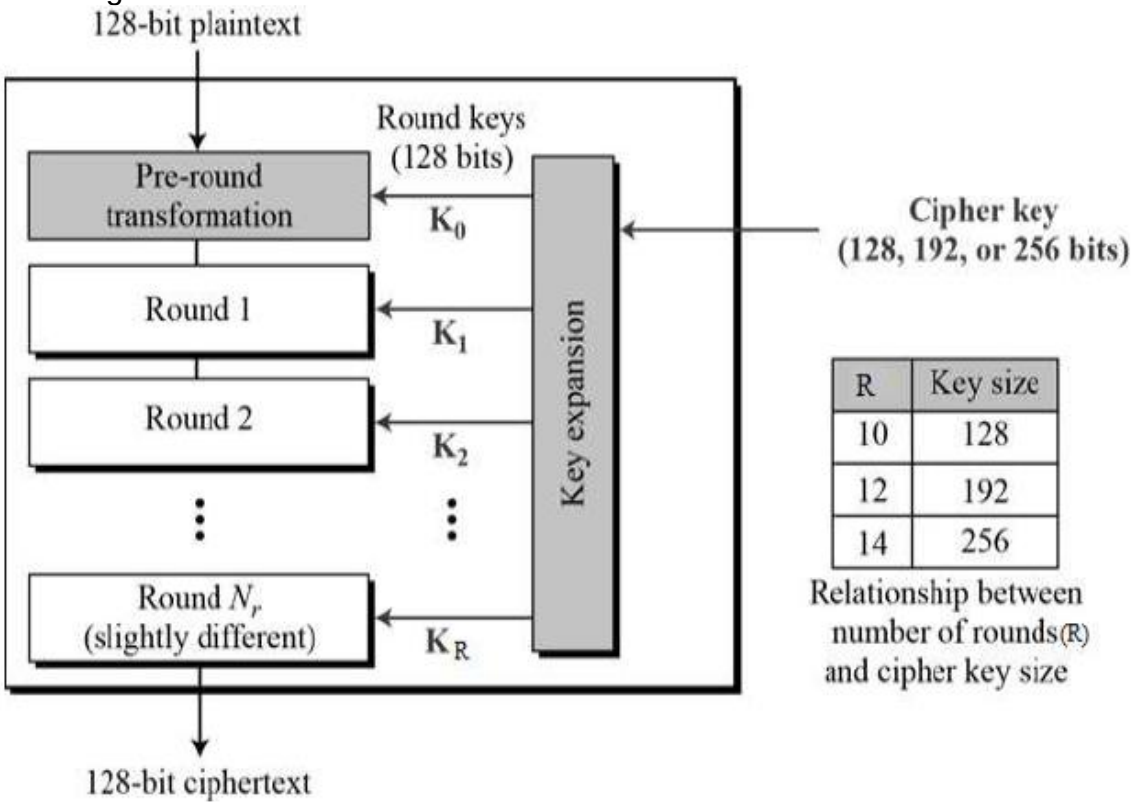
Advanced Encryption Standard

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.
A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.
The features of AES are as follows:

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

Operation of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix: Unlike DES, the number of rounds in AES is variable and depends on the length of the key.
AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. The schematic of AES structure is given in the following illustration:

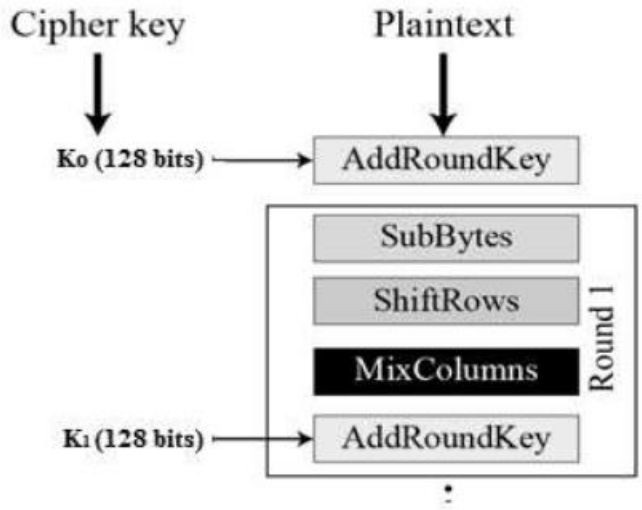


R	Key size
10	128
12	192
14	256

Relationship between number of rounds (R) and cipher key size

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise
of four sub-processes. The first round process is depicted below:



Byte Substitution (SubBytes)
The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shiftrows
Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are reinserted on the right side of row. Shift is carried out as follows:

- First row is not shifted
- Second row is shifted one (byte) position to the left
- Third row is shifted two positions to the left
- Fourth row is shifted three positions to the left
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other

MixColumns
Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Addroundkey
The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order:

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related

RSA Algorithm:

Diffie and Hellman challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems.

One of the first successful responses to the challenge was developed in 1977 by 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}. We examine RSA in this section in some detail, beginning with an explanation of the algorithm. Then we examine some of the computational and cryptanalytical implications of RSA

Description of the Algorithm

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. 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 $Med \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate $Me \bmod n$ and $Cd \bmod n$ for all values of $M < n$.
3. It is infeasible to determine d given e and n.

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 $\varphi(n)$, where $\varphi(n)$ is the Euler totient function.

For p,q prime, $\varphi(pq) = (p - 1)(q - 1)$. The relationship between e and d can be expressed as

***ed* mod φ(*n*) = 1**

This is equivalent to saying

$$ed \equiv 1 \bmod \phi(n)$$

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

That is, *e* and *d* are multiplicative inverses mod $\varphi(n)$. Note that, according to the rules of modular arithmetic, this is true only if *d* (and therefore *e*) is relatively prime to $\varphi(n)$. Equivalently, gcd($\varphi(n)$, *d*) = 1.

The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

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

Key Generation Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example of RSA algorithm