Stream and Block ciphers

Actually the Symmetric ciphers divides into two broad categories 1) Stream ciphers
2) Block ciphers.

1. Stream Ciphers:

Encryption and decryption are done typically on one symbol (character or a bit) at a time.

Let P = plaintext stream, C = ciphertext stream and k = key stream.

$P = P_1 P_2 P_3 \ldots\ldots$ $C = C_1 C_2 C_3 \ldots\ldots$, $k = (k_1, k_2, k_3, \ldots)$

$C_1 = E_{k_1}(P_1)$ $C_2 = E_{k_2}(P_2)$ $C_3 = E_{k_3}(P_3) \ldots\ldots$

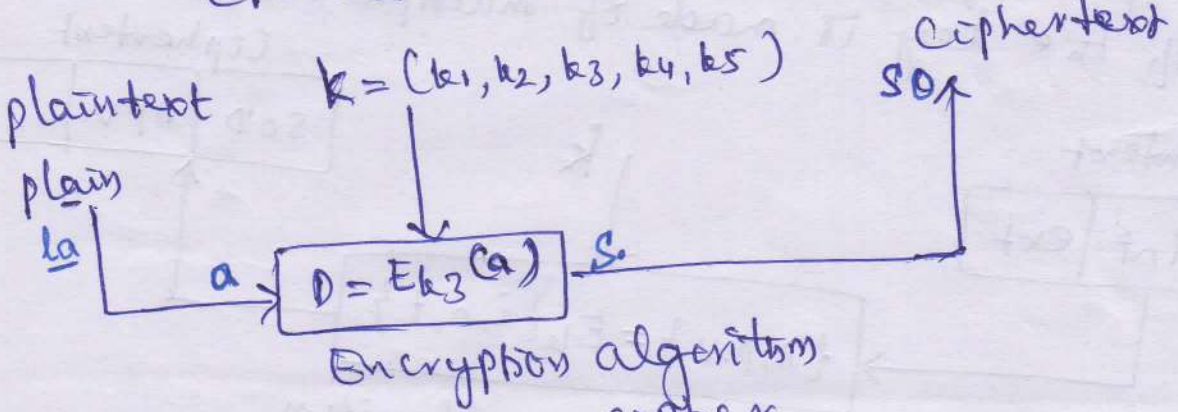


Fig (26) Stream cipher.

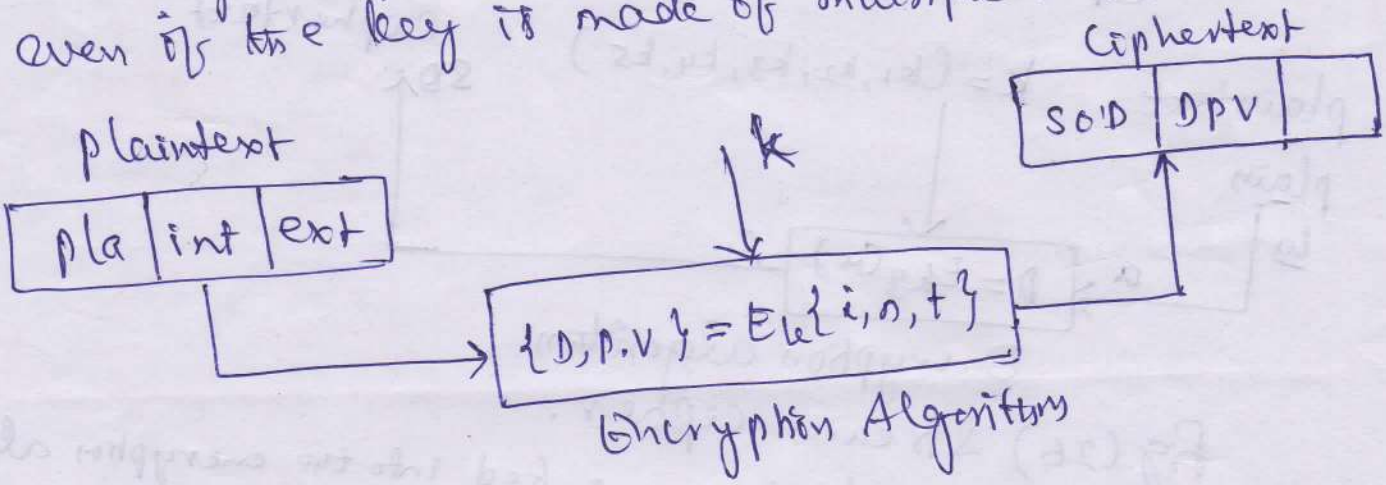
characters in the plaintext are fed into the encryption alg, one at a time; the ciphertext characters are also created one at a time. The key stream, can be created in many ways. It may be a stream of predetermined values; it may be created one value at a time using an alg.
The value may depend on the plaintext or ciphertext characters. or also depend on the previous key values.

In fig(26), third character in the plaintext stream is being encrypted using the third value in the key stream. The result creates the third character in the ciphertext stream.

Ex: Additive ciphers, the monoalphabetic substitution ciphers & Vigenere ciphers are stream ciphers.

Blocle ciphers: In a blocle cipher, a group of plaintext symbols of size m ($m > 1$) are encrypted to gether creating a group of ciphertext of the same size.

* a single key is used to encrypt the whole blocle even if the key is made of multiple values.



Fig(27) Blocle cipher

In a blocle cipher, a ciphertext blocle depends on the whole plaintext blocle.

Ex: playfair ciphers, Hill ciphers, polyalphabetic cipher are blocle ciphers.

Introduction to Modern Symmetric-key Ciphers

The traditional Symmetric-key ciphers are character-oriented ciphers. We need bit-oriented ciphers, because the information to be encrypted is not just text; it can also consist of numbers, graphics, audio & video data.

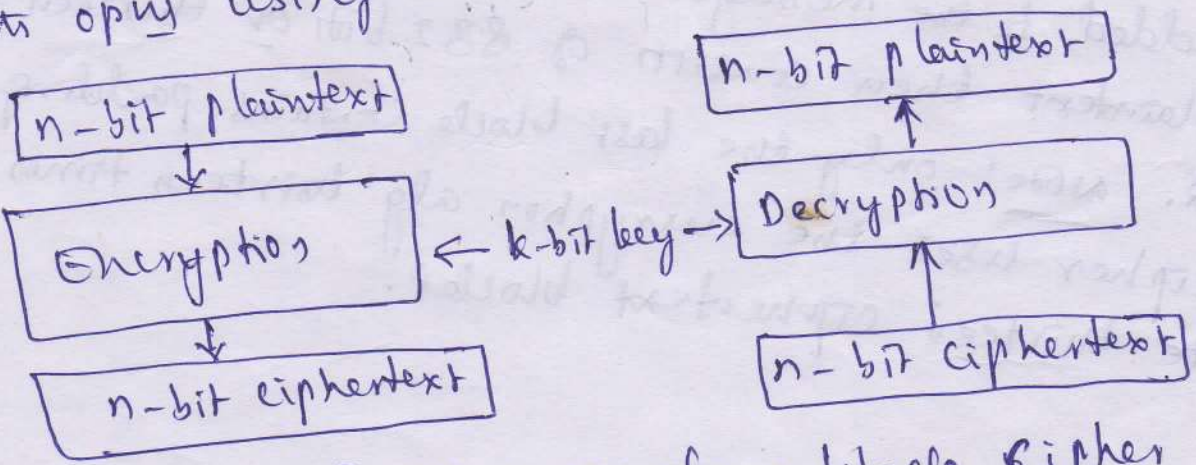
To encrypt this type of data, need to convert into a stream bits, and then to send the encrypted stream.

Modern Block ciphers

A Symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypt an n-bit block of cipher text.

The encryption or decryption algorithm uses a k-bit key.

* both opns using same secret key.



Fig(1) A modern block cipher

If the message has fewer than n bits, padding must be added to make it an n-bit block; If the message has more than n bits, it should be divided into n-bit blocks and the appropriate padding must be added to the last

block if necessary. The common values for n are 64, 128 256 or 512 bits.

Ex: How many padding bits must be added to a message of 100 characters if 8-bit ASCII is used for encoding and the block cipher accepts blocks of 64 bits?

Soln: Encoding 100 characters using 8-bit ASCII

So 800 bit message:

plaintext must be divisible by 64.

If |M| and |Pad| are the lengths of the message and the lengths of the padding.

$|M| + |Pad| = 0 \bmod 64 \rightarrow |Pad| = -800 \bmod 64$

$\rightarrow 32 \bmod 64$

i.e. 32 bits of padding (for example 0's) need to be added to the message.

The plaintext then consists of 832 bits or thirteen 64-bit blocks. Note: only the last block contains padding.

The cipher uses the encryption alg thirteen times to create thirteen ciphertext blocks.

Substitution or Transposition

A modern block cipher can be designed to act as a substitution cipher or a transposition cipher.

If the cipher is designed as a substitution cipher, a 1-bit or a 0-bit in the plaintext can be replaced by either a 0 or a 1.

i.e. plaintext and ciphertext have a different no. of 1's.

A 64-bit plaintext block of 12 0's and 52 1's can be encrypted to a ciphertext of 34 0's and 30 1's.

If the cipher is designed as a transposition cipher, the bits are only reordered (transposed); there is the same no. of 1's and 0's in the plaintext and in the ciphertext. (2m)

Modern block ciphers are designed as substitution ciphers because the inherent characteristics of transposition (preserving the no. of 1's or 0's) makes the cipher vulnerable to exhaustive-search attacks.

Ex-2: Suppose, a block cipher, $n = 64$. If there are 10 1's in the ciphertext, how many trial and error tests does Eve (attacker) need to do to recover the plaintext from the intercepted ciphertext in each of the following cases?

(a) The cipher is designed as a substitution cipher

(b) The cipher is designed as a transposition cipher

Solution: Case (a) Substitution cipher: Eve (attacker) has no idea how many 1's are in the plaintext. Eve needs to try all possible 2^{64} 64-bit blocks to find one that makes sense. If Eve could try 1 billion blocks per second, it would still take hundreds of years, on average, be she (Eve) could be successful.

Case (b) transposition cipher: Eve knows that there are exactly 10 1's in the plaintext, because transposition does not change the no. of 1's (or 0's) in the ciphertext. Eve can use exhaustive-search attack using only those 64-bit blocks that have exactly 10 1's. There are only $(64!)/[(10!)(54!)]$

$= 151\,473\,214,816$ out of 2^{64} 64 bit words that have exactly 10 1's. Eve can test all of them in less than 3 minutes if she can do 1 billion tests per second.

Block ciphers as permutation Groups

Since modern block cipher is a group: because the key is long enough to choose every possible mapping from the input to the output. This is called a full-size key cipher.

* a block cipher needs to have a key that is a secret between the sender & the receiver. In practice, the key is smaller.

1. Full-size key ciphers: Although full-size key ciphers are not used in practice.

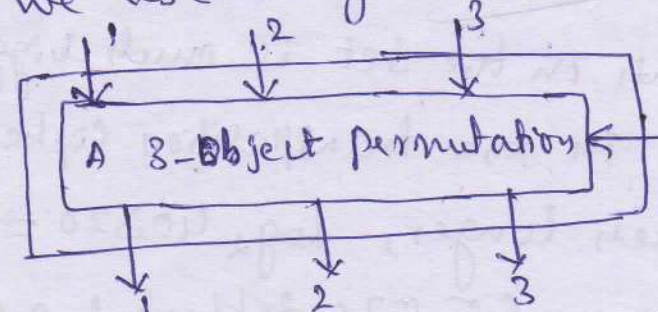
ⓐ Full-size key Transposition block ciphers:

- only transposes bits without changing their values, so it can be modeled as an n-object permutation with a set of n! permutation tables in which key defined.

If n! possible keys required, then key having $\lceil \log_2 n! \rceil$ bits

Ex: Show the model and the set of permutation tables for a 3-bit block transposition cipher where the block size is 3 bits

Soln: The set of permutation tables has 3! = 6 elements.
The key should be $\log_2 6 = 3$ bits long.
Note: a 3-bit key can select $2^3 = 8$ different mappings, we use only 6 of them.



A 3-bit block Transposition cipher

Fig(2): A transposition block cipher modeled as a permutation

{ [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2] [3, 2, 1] }
The set of permutation tables with 3! = 6 elements

⑥ Full-size key Substitution Block ciphers (it Substitutes bits)

→ a full-size key Substitution cipher can not be modeled as a permutation.

If we model the substitution cipher as a permutation, decode the input and encode the output.

Decoding means transforming an n-bit integer into a 2^n-bit string with only a single 1 and $2^n - 1$ 0's. The position of the single 1 is the value of the integer, in which the positions range from 0 to $2^n - 1$.

Encoding is the reverse process.

Because the new i/p and o/p have always a single 1, the cipher can be modeled as a permutation of $2^n!$ objects.

Ex: 4. Show the model and the set of permutation tables for a 3-bit block substitution cipher.

Solution: The three-input plaintext can be an integer between 0 to 7. This can be decoded as an 8-bit string with a single 1

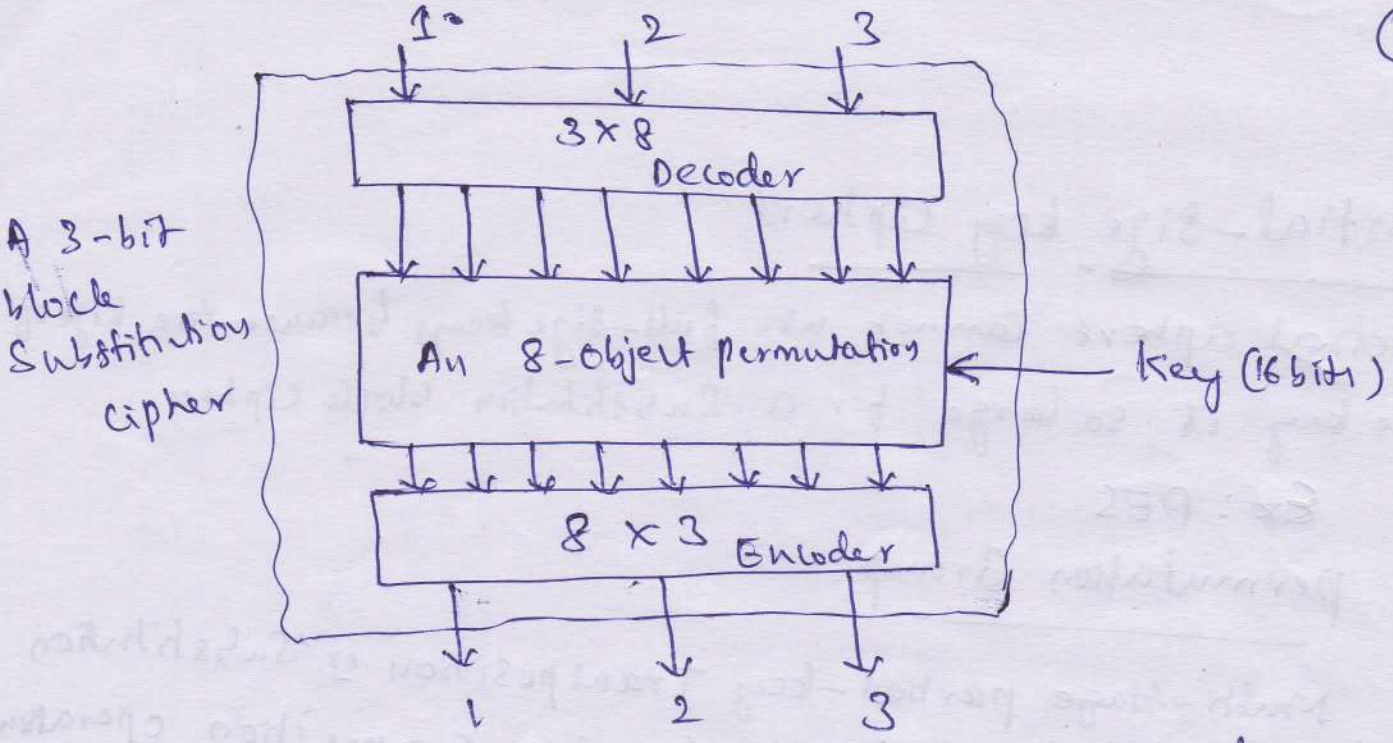
Ex: 000 can be decoded as 00000001
101 ——→ 00100000 (Refer sheet 8, back)

fig(3) shows the model and the set of permutation tables.

note: no. of elements in the set is much bigger than the no. of elements in the transposition ciphers ($8! = 40320$)

The key is also much longer, $\log_2 40320 = 16$ bits.

Although 16 bit key define 65,536 different mappings but only 40320 are used



$\{[1\ 2\ 3\ 4\ 5\ 6\ 7\ 8], [1\ 2\ 3\ 4\ 5\ 6\ 8\ 7], \ldots\ldots\}$

The set of permutation tables with $8! = 40,320$ elements

Fig (3) A Substitution block cipher model as a permutation

A full-size key n-bit transposition cipher or a substitution block cipher can be modeled as a permutation, but their key sizes are different:

for a transposition cipher, the key is $\lceil \log_2 n! \rceil$ bits long

for a substitution cipher, the key is $\lceil \log_2 (2^n)! \rceil$ bits long

Permutation Group:

* a full-size key transposition or substitution cipher is a permutation. Show that, if encryption (or decry) uses more than one stage of any of these ciphers, the result is equivalent to a permutation group under the composition operation.

② Partial-size key ciphers

Actual ciphers cannot use full-size keys because the size of the key is so large for a substitution block cipher.

Ex: DES

permutation Group!

Q. Multi-stage partial-key Transposition or Substitution is a permutation group under the composition operation.

— to achieve more security.

A partial-size-key cipher is a group if it is a subgroup of the corresponding full-size key cipher.

If the full-size key cipher makes a group

$G = \langle M, \circ \rangle$, M = set of mappings & the operation is the composition ($\circ$), then the partial size key must make a subgroup $H = \langle N, \circ \rangle$, where N is a subset of M and the opn is the same.

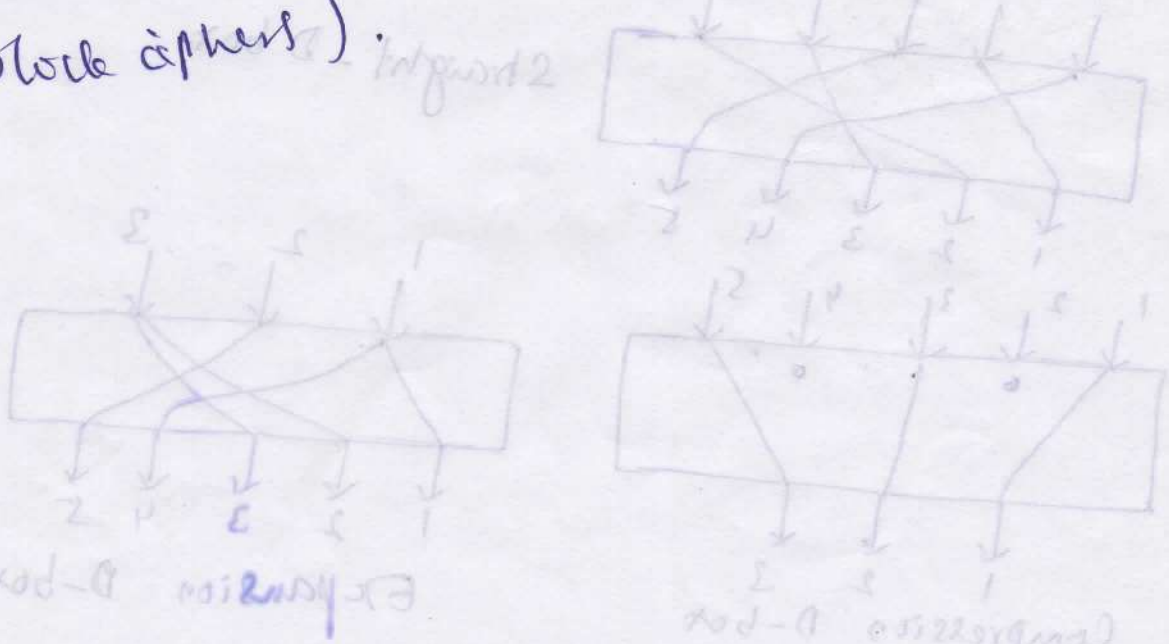
Ex: Multistage DES with a 56-bit key is not a group because no subgroup with 2^{56} mappings can be created from the corresponding group with $2^{64}!$ mappings.

"A partial-key cipher is a group under the composition operation if it is a subgroup of the corresponding full-size key cipher"

Keyless ciphers: is practically useless by itself, keyless ciphers are used as components of keyed ciphers.

① keyless Transposition ciphers: A keyless (or fixed key) transposition cipher (or unit) is like prewired transposition cipher when implemented in hardware.
A fixed ~~key~~ (single permutation rule) can be represented as a table when the unit is implemented in software.
(Ex P-boxes) → used as building blocks of modern block ciphers.

② keyless Substitution ciphers: A keyless (or fixed key) substitution cipher (or unit) is like predefined mapping from the input to output. The mapping can be defined as a table, a mathematical fn & so on.
(ex S-boxes - used as building blocks of modern block ciphers).

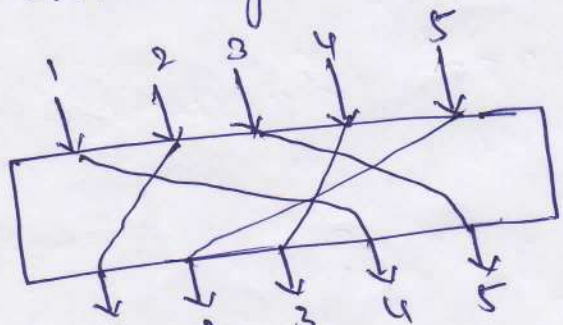


Components of a modern Block cipher

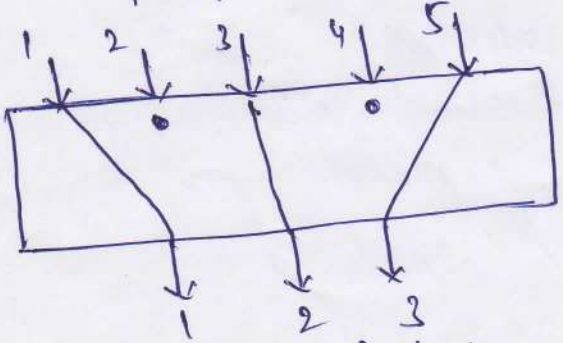
Modern block ciphers normally are keyed substitution ciphers in which the key allows only partial mappings from the possible inputs to the possible outputs. However, modern block ciphers are not designed as a single unit. To provide the required properties of a modern block cipher, such as diffusion and confusion, a modern block cipher is made of a combination of transposition units for diffusion (called D-boxes), substitution units (called S-boxes), and some other units.

① D-boxes: A D-box parallels the traditional transposition ciphers for character. It transposes bits.

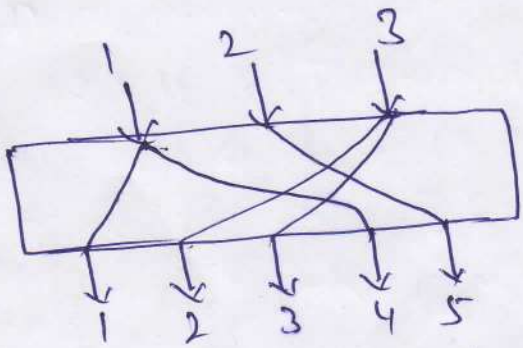
Three types of D-boxes: i) Straight D-boxes
ii) Expansion D-boxes
iii) Compression D-boxes



Straight-D box



Compression D-box



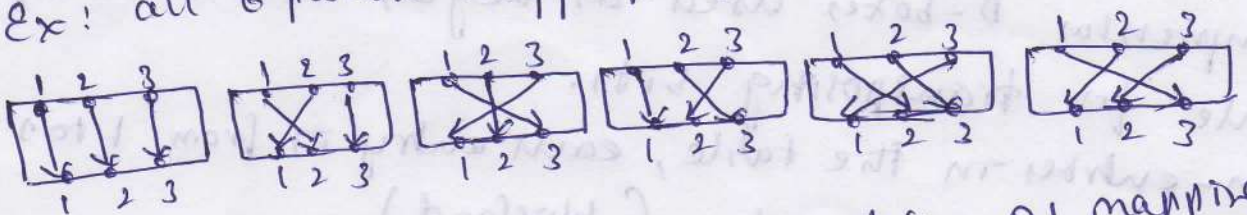
Expansion D-box

Fig(4) Three Types of D-boxes

Fig(4) shows a 5×5 straight D-box, a 5×3 compression D-box, and a 3×5 expansion D-box

Straight D-boxes : A Straight D-box with n-inputs and n-outputs is a permutation. there are n! possible mappings

Ex: all 6 possible mappings of a 3×3 D-box.



* D-box use a key to define one of the n! mappings.
* D-boxes are normally keyless. (i.e. mapping is predetermined)

If the D-box is implemented in hardware, it is prewired.

If so, a permutation table shows the rule of mapping.

i.e. the entries in the table are the inputs and the position of the entries are the o/ps.

Table 5.1 shows – a Straight permutation table when n=64

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

the position (index) of the entry corresponds to the o/p. first entry contains the no. 58, first o/p comes from the 58th input. 64th o/p from 4th input & so on.

Ex: Design an 8×8 permutation table for a Straight D-box that moves the two middle bits (bits 4 and 5) in the i/p word to the two ends (bits 1 & 8) in the o/p word. The relative positions of other bits should not be changed

Soln: need a Straight D-box with the table [4 1 2 3 6 7 8 5]

the relative positions of i/p bits 1, 2, 3, 6, 7 & 8 have not been changed.

Compression D-boxes : is a D-box with n-inputs and m-outputs where $m < n$.

* Some of the inputs are blocked.
* Compression D-boxes used as keyless with a table, rule for transposing bits.
* n-entries in the table, each entry is from 1 to n with some missing values (blocked).

Table 5.2 shows an example for a 32x24 Compression D-box. (Here, 7, 8, 9, 15, 16, 23, 24, 25 are blocked)

01	02.	03	21	22	26	27	28	29	13	14	17	
18	19	20	04	05	06	10	11	12	30	31	32	

Table 5-2 Example of a 32x24 D-box.

* need to ~~transpose~~ / permute bits & the same time decrease the no. of bits for the next stage

Expansion D-boxes : is a D-box with n-inputs and m-outputs where $m > n$. Some of the inputs are connected to more than one input, they are keyless, using table, rule for transposing bits.

- n-entries, but m-n of the entries are repeated

Table 5-3., Example of 12x16 Expansion D-box. (Here each of the inputs 1, 3, 9 & 12 is mapped to two o/ps)

01	09	10	11	12	01	02	03	03	04	05	06	07	08	09	12

Table-5.3 Example of a 12x16 D-box

* need to transpose bits and the same time increase the no. of bits for the next stage.

Invertibility: A straight D-box is invertible. This mean that we can use a straight D-box in the encryption cipher and its inverse in the decryption cipher.

The mapping defined by a straight D-box is a permutation and it is called as P-box. The permutation tables are used to inverse of each other.

Ex: How to invert a permutation table represented as a one-dimensional table

1. Original table | 6 | 3 | 4 | 5 | 2 | 1 |

6	3	4	5	2	1	2. Add indices
1	2	3	4	5	6	

3. Swap contents & indices

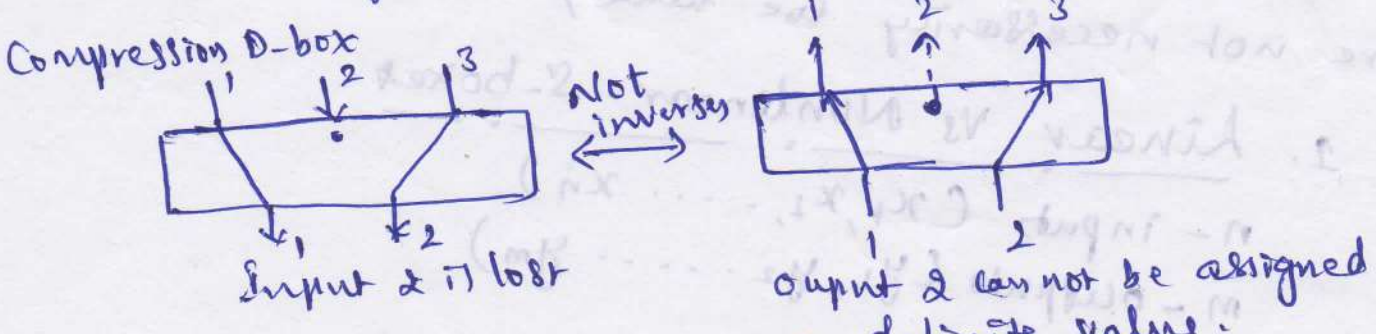
1	2	3	4	5	6
6	3	4	5	2	1

6	5	2	3	4	1	4. Sort based on indices
1	2	3	4	5	6	

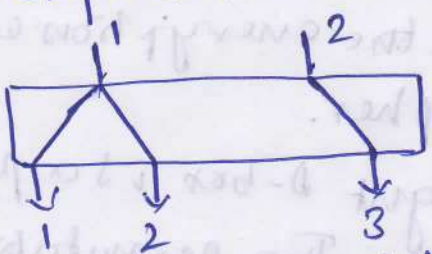
6	5	2	3	4	1

5. Inverted table

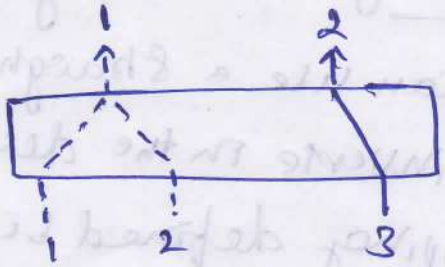
Compression and expansion D-boxes have no inverses. In a compression D-box, an input can be dropped during encryption; decryption alg does not have a clue how to replace the dropped bit. In an expansion D-box, an input may be mapped to more than one o/p during encryption; the decryption alg does not have a clue which of the several i/ps are mapped to an o/p. fig (7) demonstrates both cases.



Expansion box



Input 1 is mapped to output 1 and 2.



One of the two inputs (1 or 2) can not be selected definitely.

Fig (5.7) Compression and expansion D-boxes as non-invertible Components.

< A straight D-box is invertible, but compression & expansion D-boxes are not >

(2) S-Boxes: (like a substitution cipher), have a different number of inputs and outputs.

i.e. input to an S-box — n-bit word,
o/p — m-bit word.

* n & m are not necessarily the same.
* S-box can be keyed or keyless, modern block ciphers normally use keyless S-boxes, where the mapping from the inputs to the outputs is predetermined.

< An S-box is an m x n substitution unit, where m & n are not necessarily the same >

1. Linear Vs Nonlinear S-boxes

n - inputs $(x_1, x_2, \ldots x_n)$
m - outputs $(y_1, y_2 \ldots y_m)$

The relationship between the inputs and the o/ps can be represented as a set of equations

$$y_1 = f_1(x_1, x_2 \cdots x_n)$$
$$y_2 = f_2(x_1, x_2 \cdots, x_n)$$
$$\vdots$$
$$y_m = f_m(x_1, x_2 \cdots x_n)$$

In a linear S-box, the above relations can be expressed as

$$y_1 = a_{11}x_1 \oplus a_{12}x_2 \oplus \cdots \oplus a_{1n}x_n$$
$$y_2 = a_{21}x_1 \oplus a_{22}x_2 \oplus \cdots \oplus a_{2n}x_n$$
$$\vdots$$
$$y_m = a_{m1}x_1 \oplus a_{m2}x_2 \oplus \cdots \oplus a_{mn}x_n$$

In a non-linear S-box we can not have the above relation for every output. Such an S-box will have "and" terms, like x_1x_2, x_3x_4, etc in their expressions.

Ex: ① In an S-box with three inputs and two o/ps

$$y_1 = x_1 \oplus x_2 \oplus x_3, \quad y_2 = x_1$$

The S-box is linear because $a_{11} = a_{12} = a_{13} = a_{21} = 1$ and $a_{22} = a_{23} = 0$. The relationship can be represented by matrices

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

② In an S-box with three inputs and two o/p's,

$$y_1 = x_1x_2 \qquad y_2 = x_1 + x_2x_3$$

③ Exclusive-OR : An important component in most block ciphers is the exclusive-OR operation.

* Addition & Subtraction opns are performed by a single operation (in the GF(2^n) field) is called the ex-OR

④ Properties of EX-OR : Closure, Associativity, Commutativity, identity, Inverse.

⑤ Complement : Operation is unary opn. $x \oplus \bar{x} \cdots$

⑥ Inverse : The inverse of a component in a cipher makes sense if the component represents a unary opn. Additive inverse : $y = k \oplus k \rightarrow x = k \oplus y$

⑦ Circular shift : Another component of modern block ciphers is the circular shift operation. shifting to left or to right.

Left-shift : shift each bit on an n-bit word k-positions to the left : the left most k-bits are removed from the left and become the rightmost bits.

Right-shift : shift each bit to right, the right most k bits are removed from the right and become the leftmost bits.

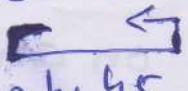
Ex : n = 8 & k = 3

Before shifting

$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$

After shifting

$b_4\ b_3\ b_2\ b_1\ b_0\ b_7\ b_6\ b_5$

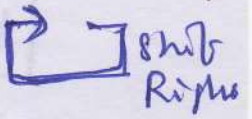


Before shifting

$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$

After shifting

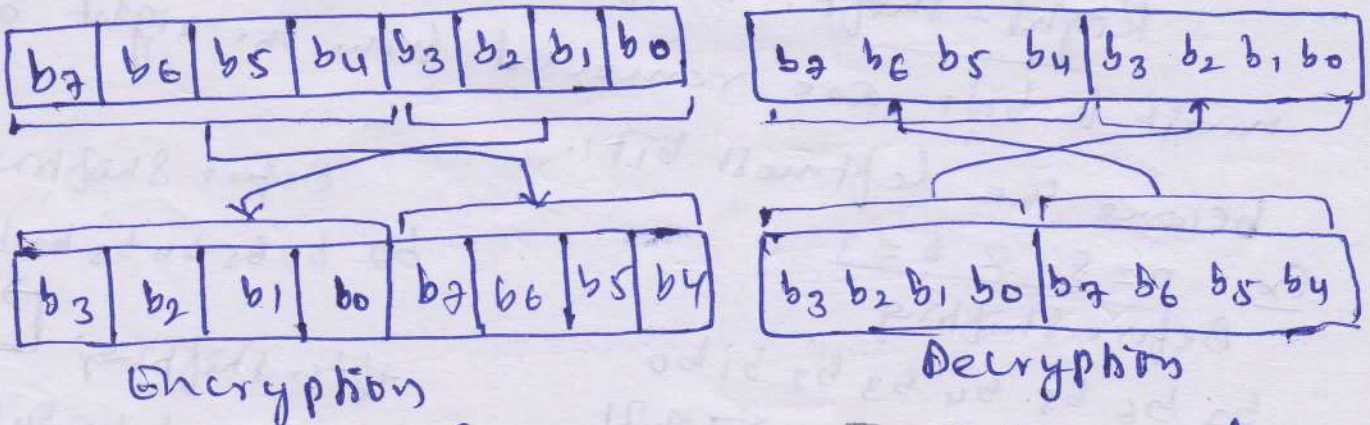
$b_2\ b_1\ b_0\ b_7\ b_6\ b_5\ b_4\ b_3$



Invertibility: A circular-left-shift opn is the inverse of the circular-right-shift opn. If one is used in encryption cipher, the other can be used in the decryption cipher.

Property: Circular-shift opn has two properties: i) shifting is modulon. i.e. If k=0 or k=n, there is no shifting. If k is larger than n, then the i/p is shifted k mod n bit. ② the circular shift opn under the composition opn is a group. i.e shifting a word more than once is the same as shifting it only once.

Swap: This opn is a special case of circular shift opn where k=n/2. i.e this opn is valid only if n is an even number. Because left shifting n/2 bit is the same as right-shifting n/2, this component is self-invertible. A swap opn in the encryption cipher can be totally cancelled by a swap opn in the decryption cipher. Fig(11) shows the swapping opn for an 8-bit word.

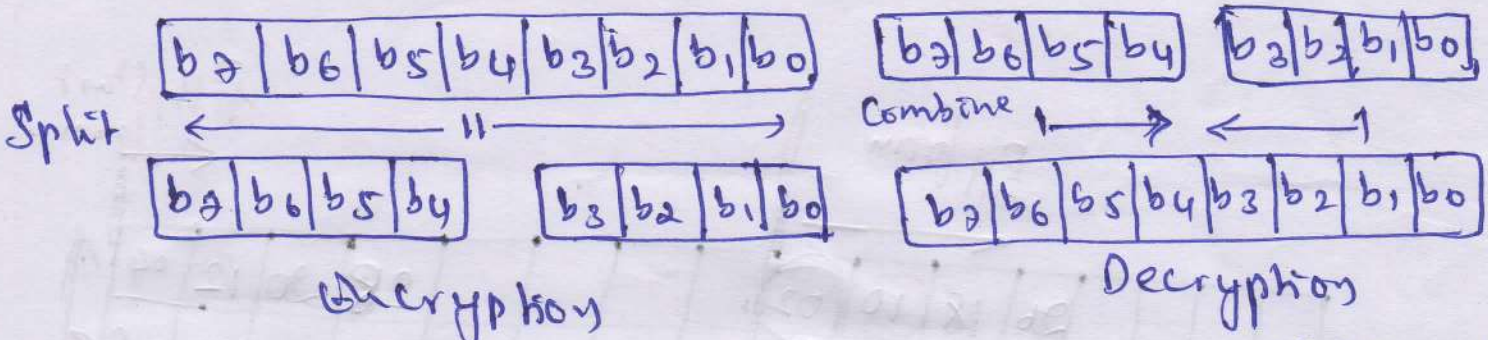


fig(11) Swap opn on an 8-bit word.

Split and Combine: The split opn normally splits an n-bit word in the middle, creating two equal-length words.

The Combine opn normally concatenates two equal-length word to create an n-bit word.

These two opns are inverses of each other and can be used as a pair to cancel each other out. If one is used in the encryption cipher, the other is used in the decryption cipher. Fig(12) shows the two opns in the case where n = 8



Fig(12) Split and Combine opns on an 8-bit word

Product ciphers: Shanon introduced the concept of a product cipher. A product cipher is a complex cipher combining substitution, permutation and other components

@ **Diffusion and Confusion**: The idea of these two properties is to hide the ciphertext & plaintext, key The diffusion is to hide the relationship between the ciphertext and plaintext. This will frustrate the adversary who uses ciphertext statistics to find the plaintext. Diffusion implies that each symbol/character in the ciphertext is dependent on some or all symbols in the plaintext.

The idea of Confusion is to hide the relationship betn the ciphertext & key. This will frustrate the adversary who tries to use the ciphertext to find the key. In other word, If a single bit in the key is changed most or all the bits in the ciphertext will also be changed.

⑤ **Rounds**: Diffusion and confusion can be achieved using iterated product ciphers where each iteration is a combination of S-boxes, P-boxes and other-components. Each iteration is referred to as a round. The block cipher uses a key schedule or key generator that creates different keys for each round from the cipher key.

In an N-round cipher, the plaintext is encrypted N. times to create the ciphertext; the ciphertext is decrypted N times to create the plaintext.

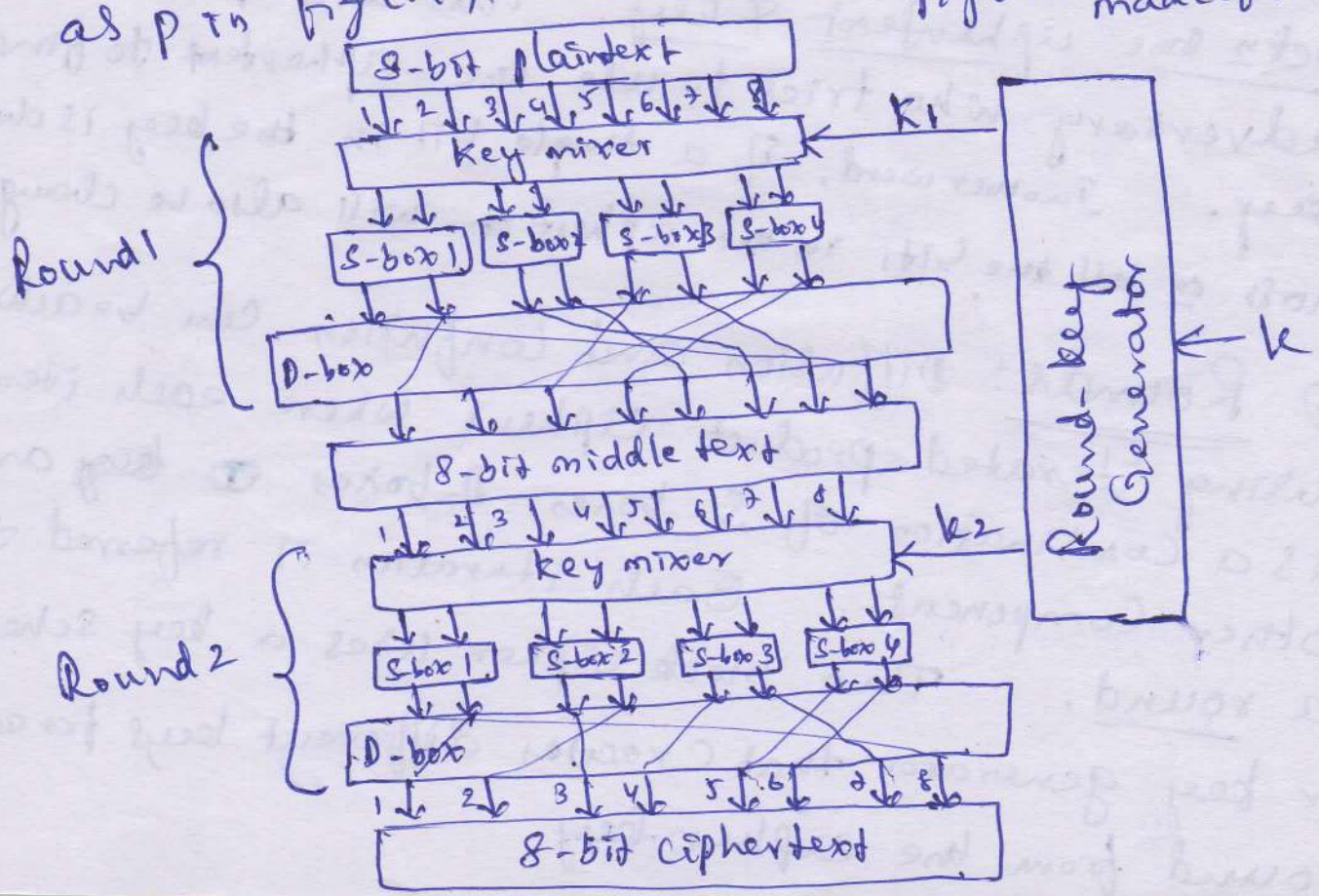
Fig (13) shows a simple product cipher with two rounds. Three transformations happen at each round.

(a) the 8-bit text is mixed with the key, done by ex-OR the 8-bit word with the 8-bit key.

(b) the o/ps of the above organized into four 2-bit groups and are fed into four S-boxes. The values of bits are changed based on the structure of the S-boxes

(c) the o/ps of S-boxes are passed through a P-box to transpose the bits so that in the next round each box receives different inputs. Since this is a straight P-box and has to be invertible (for the purpose of decryption), so it is referred to as a permutation (P-box) and is denoted as P in fig (14).

fig (13) A product cipher made of two rounds



Diffusion: The primitive design of fig(13) shows how a product with the combination of S-boxes and P-boxes can guarantee diffusion. fig(14) shows how changing a single bit in the plaintext affects many bits in the ciphertext.

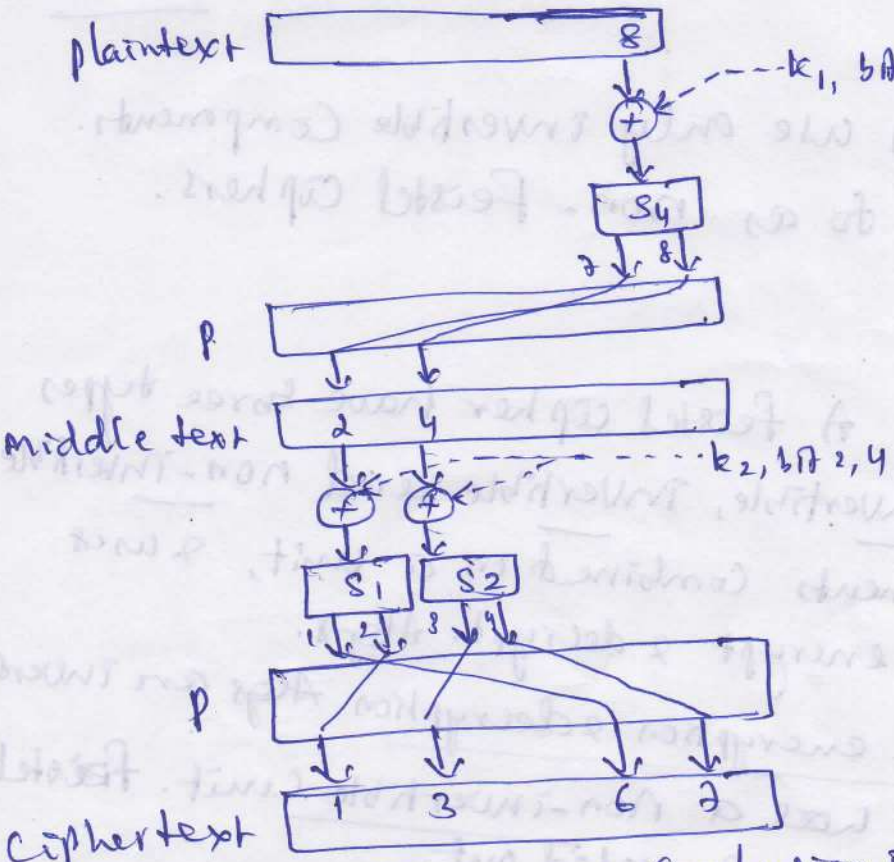


Fig (14) Diffusion & Confusion in a block cipher.

Confusion: Fig(14) also shows how the confusion property can be achieved through the use of a product cipher.

Practical ciphers: To improve diffusion and confusion, practical ciphers use large data blocks, more S-boxes and more rounds.

Two classes of product ciphers

The first class ciphers use both invertible and non-invertible components. Ciphers in this class referred to as **Feistel ciphers**. Ex: DES

The second class ciphers use only invertible components. This class referred to as non-Feistel ciphers.

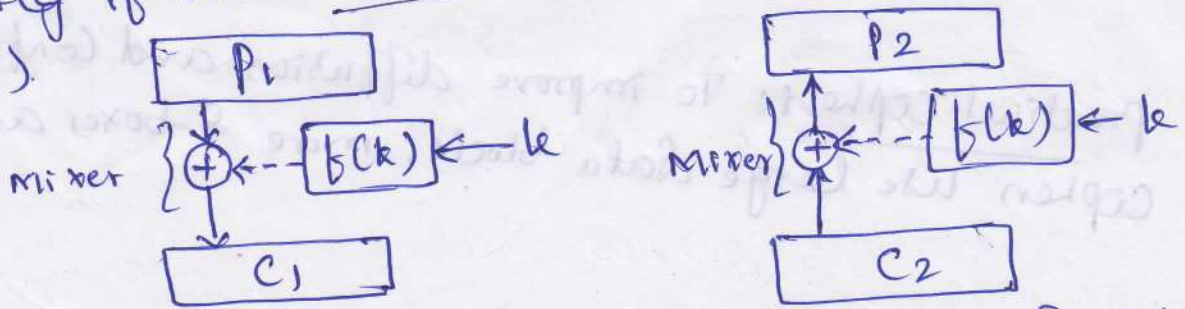
Ex: AES

Feistel Ciphers: A feistel cipher have three types of components: Self-invertible, invertible and non-invertible.

- all non-invertible elements combined in a unit, & uses the same unit in encrypt & decrypt. Algs.

The question is how the encryption & decryption Algs are inverses of each other if each has a non-invertible unit. Feistel showed that they can be canceled out.

First Thought: To better understand the feistel cipher, let us consider the same non-invertible component in the encryption and decryption algs. The effects of a non-invertible component in the encrypt alg can be canceled in the decrypt. Alg if we use an exclusive-OR op^n, as shown in fig(15).



Fig(15) The first thought in feistel cipher design.

In the encryption, a non-invertible function, f(k), accepts the key as input. The o/p of this component is Ex-ORed with the plaintext. The result becomes the ciphertext.

- Combination of the fn & Ex-OR opn → mixer

The mixer plays an important role in the later development of the Feistel cipher.

Because the key is the same in encryption and decryption we can prove that the two algs are inverses of each other.

i.e. If $C_2 = C_1$ (no change in the ciphertext during transmission then), $P_2 = P_1$

Encryption: $C_1 = P_1 \oplus f(k)$

Decryption: $P_2 = C_2 \oplus f(k) = C_1 \oplus f(k) = P_1 \oplus f(k) \oplus f(k)$

$= P_1 \oplus (00..0) = P_1$

Note: Two properties of Ex-OR opn have used (existence of inverse & identity).

The above argument proves that, the mixer itself is self-invertible.

(The mixer in the Feistel design is self-invertible)

Ex: The plaintext & ciphertext are each 4-bits long and the key is 3bits long. Assume that the fn takes the first and third bits of the key, interprets these two bits as a decimal number, squares the number, and interprets the result as a 4-bit binary pattern. Show the results of encryption and decryption of the original plaintext is 0111 and the key is 101.

Encryption: $c = p \oplus f(k) = 0111 \oplus$ solution (P 50).

Solution: The fn extracts the first and second bits to get 11 in binary or 3 in decimal. The result of squaring as 9, which is 1001 in binary.

Encryption: $C = P \oplus f(k) = 0111 \oplus 1001 = 1110$

Decryption: $P = C \oplus f(k) = 1110 \oplus 1001 = 0111$ same as the original P

The function $f(101) = 1001$ is non-invertible, but the Ex-OR operation allows to use the fn in both encryption and decryption algs. (The fn is non-invertible, but the mixer is self-invertible)

Improvement:

* Input to the fn is also a part of the plaintext and ciphertext.

The key can be used as the second input to the function. The function can be a complex element with some keyless and some keyed elements.

To achieve this goal, divide the plaintext and the ciphertext into two equal-length blocks, left (L) and right (R). Let the right block be the input to the fn and let the left block be ex-ored with the fn o/p.

(note: the inputs to the fn must be exactly the same in the encryption & decryption).

Right sections of the plaintext & ciphertext are equal. In other words, the right section go into & come out unchanged.

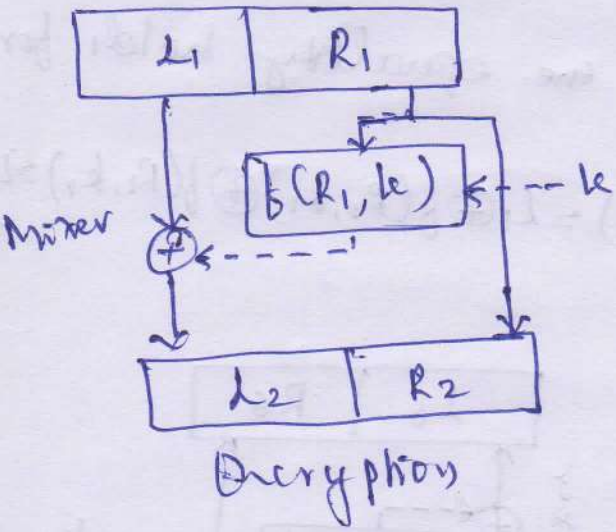
Fig(16) shows the Idea.

The encryption & decryption Algs are still inverses of each other. Assume that $L_3 = L_2$ and $R_3 = R_2$ (no change in the ciphertext during transmission).

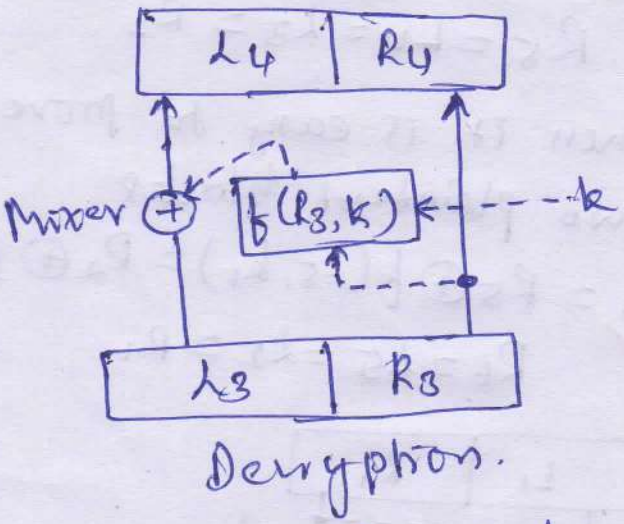
$L_2 = L_1 + f(R_1, k)$

$R_4 = R_3 = R_2 = R_1$

$L_4 = L_3 \oplus f(R_3, k) = L_2 \oplus f(R_2, k) = L_1 \oplus f(R_1, k) \oplus f(R_1, k) = L_1$ $\because R_2 = R_1$



Encryption



Decryption.

Fig (16) Improvement of the previous feistel design.

The plaintext used in the encryption alg is correctly regenerated by the decryption Alg.

Final Design: (Drawback of previous design: one flaw, right half never changes, eve find right half ------->

To improve the previous design, ① first increase the no. of rounds, ② second, add a new element to each round: a swapper. The effect of swapper in the encrypt & decrypt is canceled. However, it allows to swap the left & right halves in each round. Fig (17) shows the new design with two rounds.

Note: Two rounds of keys k_1, k_2 are used in inverse order

Two mixers are inverses of each other

Swappers are inverses of each other

So it should be clear that the encryption & decrypt ciphers are inverses of each other.

Let $L_0 = L_1$ & $R_0 = R_1$, Assuming $L_4 = L_3$ & $R_4 = R_3$

Prove the equality for the middle text.

$R_5 = R_4 \oplus f(L_4, K_2) = R_3 \oplus f(R_2, k_2) = L_2 \oplus f(R_2, k_2)$

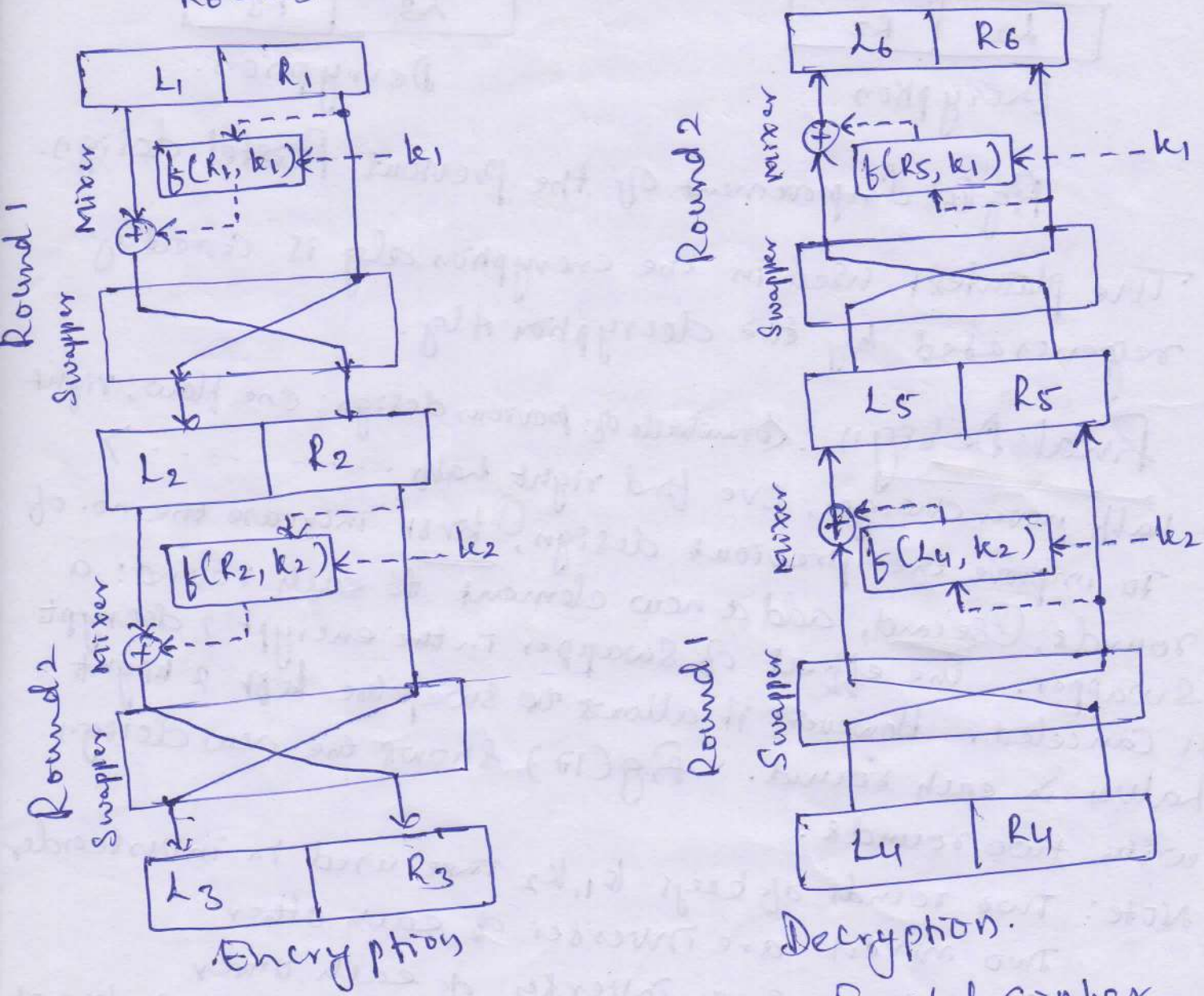
$\oplus f(R_2, k_2) = L_2$

$R_5 = L_4 = L_3 = R_2$

Then it is easy to prove that the equality holds for two plaintext blocks

$L_6 = R_5 \oplus f(L_5, k_1) = R_2 \oplus f(L_2, k_1) = L_1 \oplus f(R_1, k_1) \oplus f(R_1, k_1) = L_1$

$R_6 = L_5 = L_2 = R_1$



Encryption

Decryption.

Fig(17) final Design of a feistel cipher with two rounds.

Non-feistel Ciphers uses only invertible components.

A component in the plaintext has the corresponding component in the cipher.

Ex: S-boxes: Equal no. of inputs & outputs.

- No need to divide the plaintext into two halves

Fig(13) is a non-feistel cipher.

Because components in each round are Ex-OR (self-invertible), 2x2 S-boxes, Straight P-box,

we only need to use the round keys in the reverse order. The encryption uses round keys k_1 & k_2, the decryption alg needs to use round keys k_2 and k_1

Attacks Designed for Block ciphers

Attacks on block-ciphers that are based on the structure of the modern block ciphers. These attacks use differential and linear cryptanalysis techniques.

① **Differential Cryptanalysis** — this is a chosen-plaintext attack; by accessing computer, submitting chosen plaintext and obtaining ciphertext. The goal is to find cipher key.

Algorithm analysis: Before using the chosen plaintext attacks, first analyze the encryption alg in order to collect some info about plaintext-ciphertext relationships.

Actually attacker (Eve) does not know the cipher key. However, some ciphers have weakeness in their structure that can allow Eve to find a relationship between the plaintext differences and ciphertext differences leaking info about the key.

Launching a chosen-plaintext Attack

After the analysis, which can be done once and kept for future uses as long as the structure of the cipher does not change. Eve/attacker can choose the plaintexts for attacker. The differential probability distribution table (Table 5-5) helps to choose plaintext that have the highest probability in the table.

Guessing the key value: After launching some attacks with appropriate chosen plaintexts, Eve can find some plaintext-ciphertext pairs that allow to guess the value of the key. This step starts from C and makes toward P.

General procedure: Actually modern block ciphers are more complex, they are made from different rounds. Eve/Attacker can use the following strategy:

① Because each round is the same: Eve can create a differential distribution table (XOR profile) for each S-box & combine them to create the distribution for each round.

(2) Assuming that each round is independent (a fair assumption), Eve can create a distribution table for the whole cipher by multiplying the corresponding probabilities

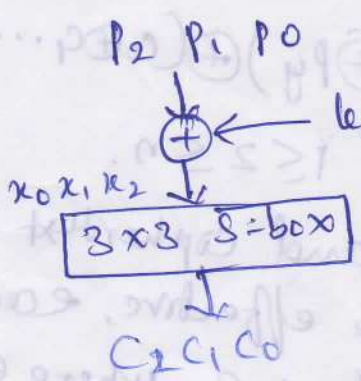
(3) Eve can make a list of plaintext for attacks based on the distribution table in Step2.

{ "Differential cryptanalysis is based on a non-uniform differential distribution table of the S-boxes in a block cipher" }

(2) Linear Cryptanalysis:

The analysis uses known-plaintext attacks.

idea: Assume that the cipher is made of a single round, as shown in fig(20). $C_0\ C_1\ C_2$ – three bits o/p and $x_0\ x_1\ x_2$ – three bits input of the S-box.



$C_0 = x_0 \oplus x_1$

$C_1 = x_0 \oplus x_1 \oplus x_2$

$C_2 = x_1 \oplus x_2$

Fig(20) A Simple cipher with a linear S-box

The S-box is a linear transformation, with this linear component, we can create three linear equations between plaintext and ciphertext bits

$C_0 = P_0 \oplus k_0 \oplus P_1 \oplus k_1$

$C_1 = P_0 \oplus k_0 \oplus P_1 \oplus k_1 \oplus P_2 + k_2$

$C_2 = P_1 \oplus k_1 \oplus P_2 \oplus k_2$

Solving for three unknowns, we get

$$k_1 = (P_1) \oplus (C_0 \oplus C_1 \oplus C_2)$$
$$k_2 = P_2 \oplus (C_0 \oplus C_1)$$
$$k_0 = P_0 \oplus (C_1 \oplus C_2)$$

This means that a single known-plaintext value can find the values of k_0, k_1 and k_2. However, real block ciphers are not as simple as this one; they have more components and the S-boxes are not linear.

Linear Approximation: In some modern ciphers, it may happen that some S-boxes are not totally nonlinear; they can approximated, probabilistically, by some linear eqns.

In general, Given a cipher with plaintext and ciphertext of n bits and a key of m bits. So some eqns of the form:

$$(k_0 \oplus k_1 \oplus \cdots \oplus k_x) = (P_0 \oplus P_1 \oplus \cdots \oplus P_y) \oplus (C_0 \oplus C_1 \cdots + C_z)$$

where $1 \le x \le m$, $1 \le y \le n$ and $1 \le z \le n$.

The bits in the intercepted plaintext and ciphertext can be used to find the key bits. To be effective, each eqn should hold with probability $1/2 \pm \epsilon$, where ϵ is called the bias. An equation with larger ϵ is more effective than one with smaller ϵ.

Data Encryption Standard (DES)

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

History: In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem. A proposal from IBM, a modification of a project was accepted as DES. DES was published in the Federal Register in March 1975 as a draft of the **Federal Information Processing Standard (FIPS).**

After the publication, the draft was criticized severely for two reasons.

First, critics questioned the small key length (only 56 bits), which could make the cipher vulnerable to brute-force attack.

Second, critics were concerned about some hidden design behind the internal structure of DES. That allows the **National Security Agency (NSA)** to decrypt the messages without the need for the key.

Later IBM designers mentioned that the internal structure was designed to prevent differential cryptanalysis.DES was finally published as FIPS 46 in the Federal Register in January 1977.

NIST defines DES as the standard for use in unclassified applications. DES has been the most widely used symmetric-key block cipher since its publication.

NIST later issued a new standard (FIPS 46-3) that recommends the use of triple DES (repeated DES cipher three times) for future applications.

Overview: DES is a block cipher, as shown in Figure below.

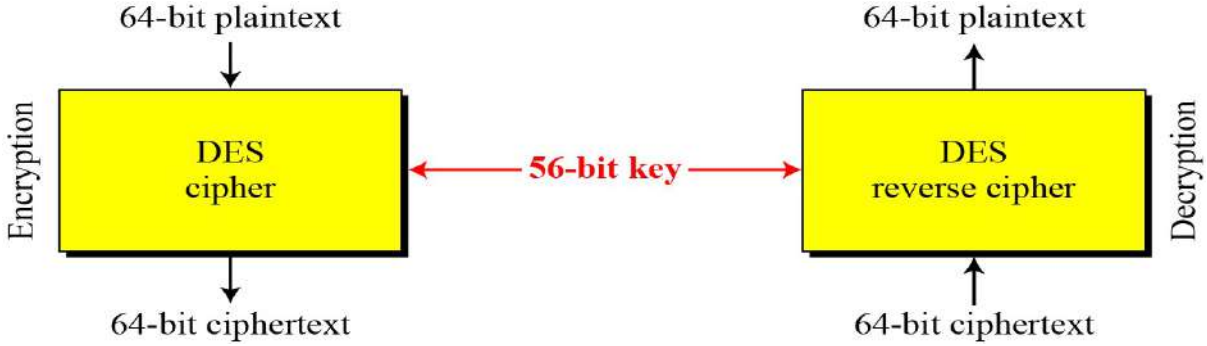


Figure: Encryption and decryption with DES

At the encryption site, DES takes a 64-bit plaintext and creates a 64-bit ciphertext;

At the decryptionsite, DES takes a 64-bit ciphertext and creates a 64-bit block of plaintext.

The same 56-bit cipher key isused for both encryption and decryption.

DES STRUCTURE

The encryption process is made oftwo permutations (P-boxes), **initial and final permutations, and sixteen Feistelrounds**.Each round uses a different 48-bit round key generated from the cipher key according to a predefinedalgorithm. Figure 2 shows the elements of DES cipher at the encryptionsite.

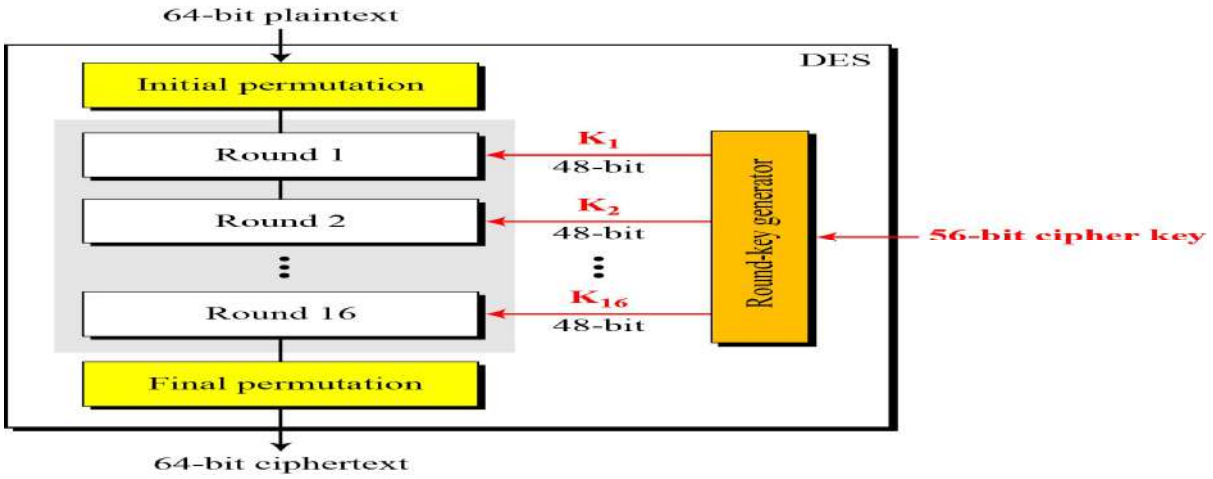


Figure 2 General structure of DES

Initial and Final Permutations

Figure 3 shows the initial and final permutations (P-boxes). Each of these permutations takes a 64-bitinput and permutes them according to a predefined rule. These permutations are keyless straight permutations that are the inverse ofeach other. For example, in the initial permutation, the 58thbit in the input becomes the first bit in theoutput. Similarly, in the final permutation, the first bit in the input becomes the 58th bit in the output.

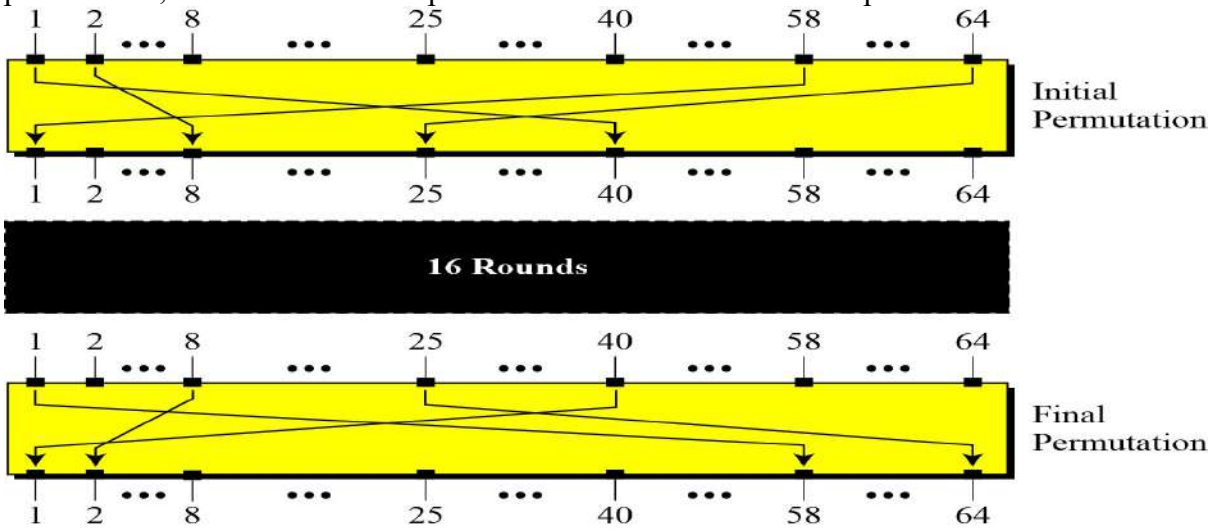


Figure 3 Initial and final permutation steps in DES

The permutation rules for these P-boxes are shown in Table 1. Each side of the table can be thoughtof as a 64-element array. Note that the valueof each element defines the input port number, and the order (index) of the element defines the output port number.

Table 1 Initial and final permutation tables

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

These two permutations have no cryptography significance in DES. Both permutations are keylessand predetermined. The reason they are included in DES is not clear and has not been revealed by theDES designers. The guess is that DES was designed to be implemented in hardware (on chips) and thatthese two complex permutations may stop a software simulation of the mechanism.

Example 1 Find the output of the initial permutation box when the input is given in hexadecimalas:0x0002 0000 0000 0001

Solution The input has only two 1^s (bit 15 and bit 64); the output must also have only two 1^s (thenature of straight permutation). Using Table 1, we can find the output related to these two bits. Bit 15in the input becomes bit 63 in the output. Bit 64 in the input becomes bit 25 in the output. So the outputhas only two 1s, bit 25 and bit 63.

The result in hexadecimal is0x0000 0080 0000 0002

Example 2 Prove that the initial and final permutations are the inverse of each other by finding the output of the final permutation if the input is0x0000 0080 0000 0002

Solution: Only bit 25 and bit 64 are 1s; the other bits are 0s. In the final permutation, bit 25 becomes bit 64 and bit 63 becomes bit 15. The result0x0002 0000 0000 0001

"The initial and final permutations are straight D-boxes that are inverses of each other and hence are permutations. They have no cryptography significance in DES"

Rounds

DES uses 16 rounds. Each round of DES is a Feistel cipher, as shown in Figure 4. The round takes L_{I-1} and R_{I-1} from previous round (or the initial permutation box) and creates L_I and R_I, which go to the next round (or final permutation box). Assume that each round has two cipher elements (mixer and swapper). Each of these elements is invertible.

The swapper is obviously invertible. It swaps the left half of the text with the right half. The mixer is invertible because of the XOR operation. All noninvertible elements are collected inside the function **f**(R_{I-1}, K_I).

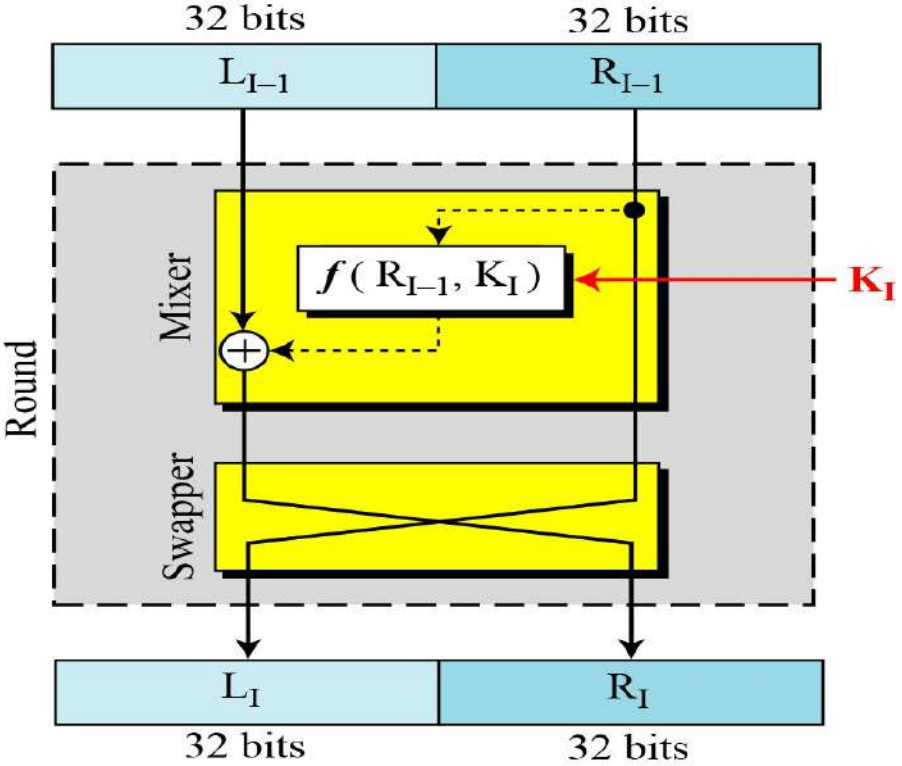


Figure 4 A round in DES (encryption site)

DES Function: The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits (R_{I-1}) to produce a 32-bit output. This function is made up of four sections: an expansion D-box, a whitener (that adds key), a group of S-boxes, and a straight D-box as shown in Figure 5.

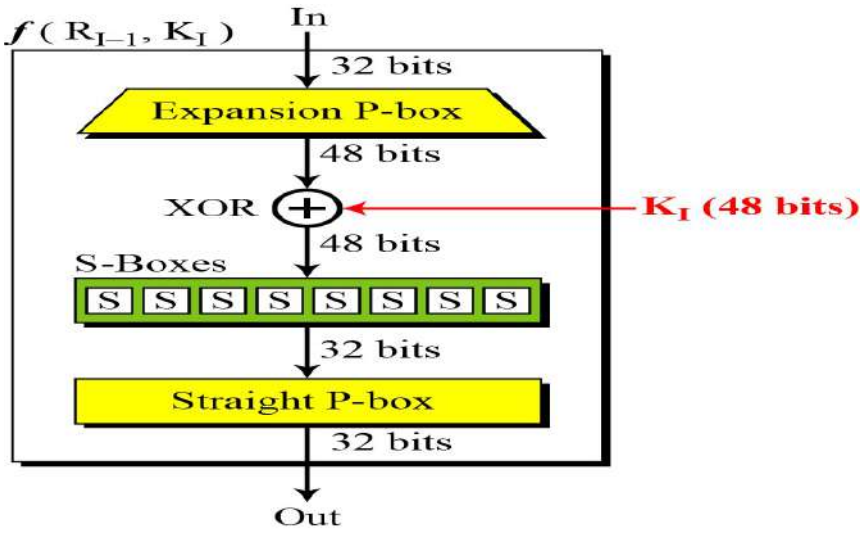


Figure 5 DES function

Expansion D-box: Since R_{I-1} is a 32-bit input and $K_I$is a 48-bit key, first need to expand R_{I-1} to48 bits. R_{I-1} is divided into 8 4-bit sections. Each 4-bitsection is then expanded to 6 bits. This expansion permutation follows a predetermined rule. For each section, input bits 1, 2, 3, and 4 are copied to output bits 2, 3, 4, and 5, respectively. Output bit 1 comes from bit 4 of the previous section; output bit 6 comes from bit 1 of the next section. If sections 1 and 8 can be considered adjacent sections, the same rule applies to bits 1 and 32. Figure 6 shows the input and output in the expansion permutation.

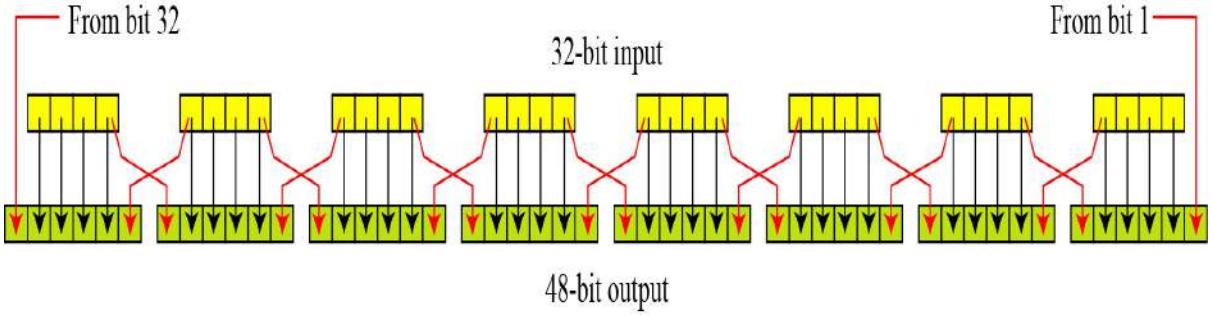


Figure 6 Expansion Permutation

Although the relationship between the input and output can be defined mathematically, DES usesTable 2 to define this D-box. Note that the number of output ports is 48, but the value range is only1 to 32. Some of the inputs go to more than one output. For example, the value of input bit 5 becomesthe value of output bits 6 and 8.

Table 2 Expansion D-box table

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

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

S-Boxes: The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. See Figure 7.

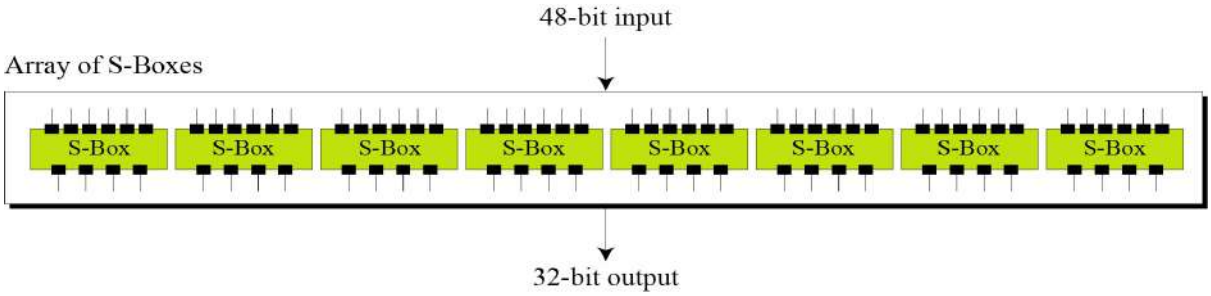


Figure 7 S-boxes

The 48-bit data from the second operation is divided into eight 6-bit chunks, and each chunk is fed into a box. The result of each box is a 4-bit chunk; when these are combined the result is a 32-bit text. The substitution in each box follows a pre-determined rule based on a 4-row by 16-

column table. The combination of bits 1 and 6 of the inputdefines one of four rows; the combination of bits 2 through5 defines one of the sixteen columns as shown in Figure 8.This will become clear in the examples.

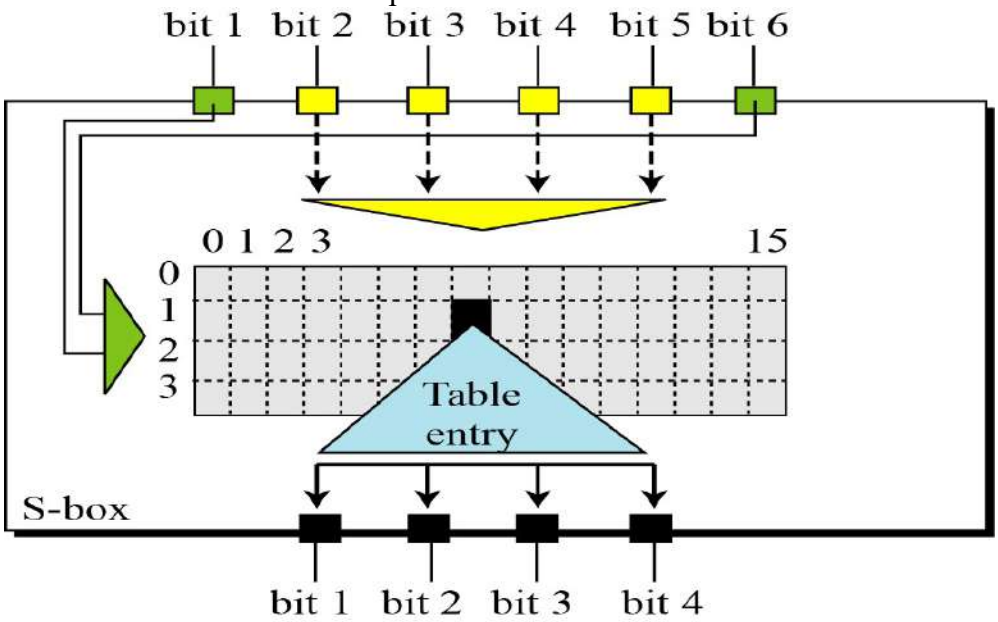


Figure 8 S-box rule

Because each S-box has its own table, need eight tables, as shown in Tables 3 to 10, to define the output of these boxes. The values of the inputs (row number and column number) and the values of the outputs are given as decimal numbers to save space. These need to be changed to binary.

Table 6.3 *S-box 1*

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

Table 6.4 *S-box 2*

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

Table 6.5 *S-box 3*

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

Table 6.6 *S-box 4*

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

Table 6.7 *S-box 5*

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

Table 6.8 *S-box 6*

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

Table 6.9 *S-box 7*

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

Table 6.10 *S-box 8*

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

Example 3 The input to S-box 1 is **1**00011. What is the output?

Solution If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal.The remaining bits are 0001 in binary, which is 1 in decimal. We look for the value in row 3, column 1,in Table 3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input 100011 yieldsthe output 1100.

Example 4 The input to S-box 8 is **0**00000. What is the output?

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

Final Permutation: The last operation in the DES function is a permutation with a 32-bit input and a 32-bit output. The input/output relationship for this operation is shown in Table 11 and follows thesame general rule as previous tables. For example, the seventh bit of the input becomes the second bitof the output.

Table 11 Straight permutation table

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

Cipher and Reverse Cipher

Using mixers and swappers, we can create the cipher and reverse cipher, each having 16 rounds. Thecipher is used at the encryption site; the reverse cipher is used at the decryption site. The whole idea isto make the cipher and the reverse cipher algorithms similar.

First Approach: To achieve this goal, one approach is to make the last round (round 16) differentfrom the others; it has only a mixer and no swapper. This is done in Figure 9.

Although the rounds are not aligned, the elements (mixer or swapper) are aligned. The final and initial permutations are also inverses of each other. The left section of the plaintext at the encryption site, L_0, is enciphered as L_{16}atthe encryption site; L_{16} at the decryption is deciphered as L_0 at the decryption site. The situation is the same with R_0 and R_{16}.

A very important point we need to remember about the ciphers is that the round keys (K_1 to K_{16})should be applied in the reverse order. At the encryption site, round 1 uses K_1 and round 16 uses K_{16}; atthe decryption site, round 1 uses K_{16} and round 16 uses K_1.

In the first approach, there is no swapper in the last round.

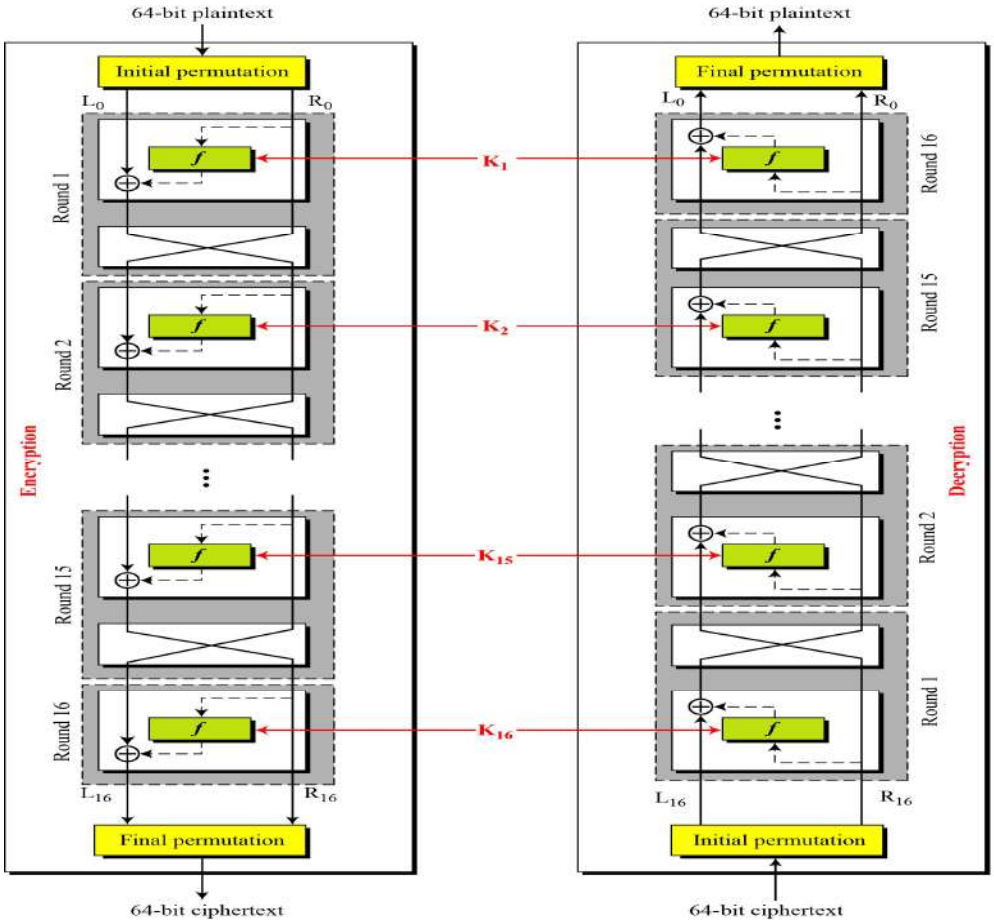


Figure 9 DES cipher and reverse cipher for the first approach

Alternative Approach: In the first approach, round 16 is different from other rounds; there is no swapper in this round. This is needed to make the last mixer in the cipher and the first mixer in the reverse cipher aligned. We can make all 16 rounds the same by including one

swapper to the 16throundand add an extra swapper after that (two swappers cancel the effect of each other). We leave the design for this approach as an assignment activity.

Key Generation: The **round-key generator** creates sixteen 48-bit keys out of a 56-bit cipher key. However, the cipher key is normally given as a 64-bit key in which 8 extra bits are the parity bits, whichare dropped before the actual key-generation process, as shown in Figure10.

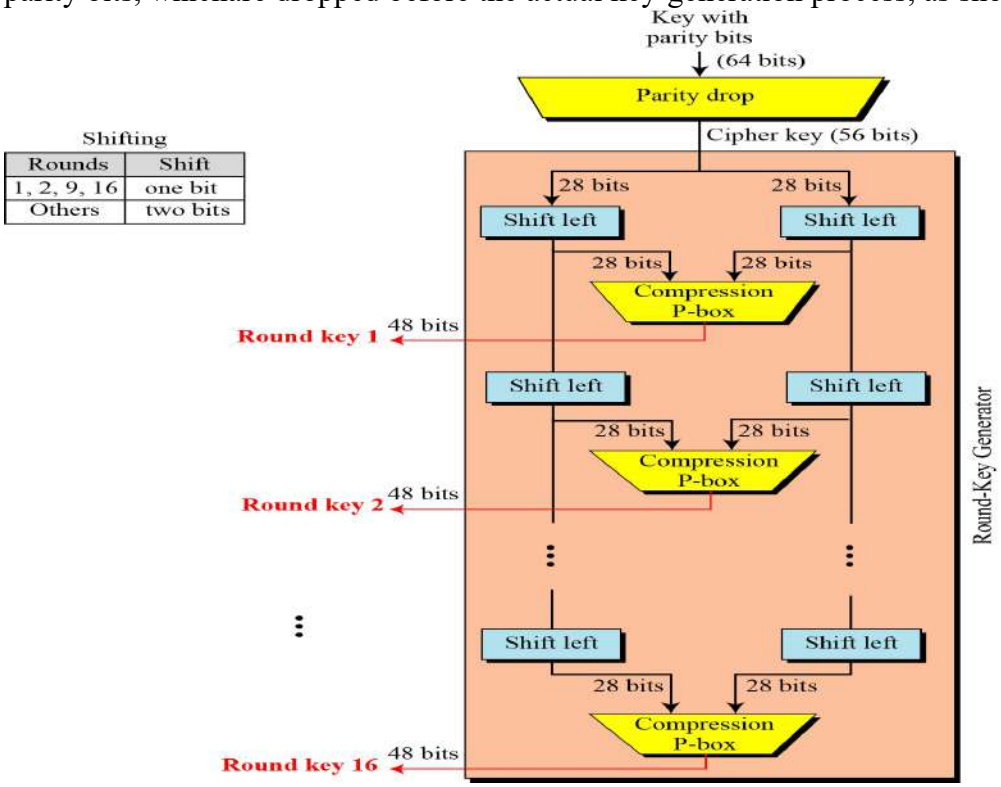


Figure 10 Key generation

Parity Drop The preprocess before key expansion is a compression transposition step that we call **parity bit drop.** It drops the parity bits (bits 8, 16, 24, 32, …, 64) from the 64-bit key and permutes the rest of the bits according to Table 6.12. The remaining 56-bit value is the actual cipher key which is used to generate round keys. The parity drop step (a compression D-box) is shown in Table 6.12.

Table 6.12 Parity-bit drop table

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

Shift Left After the straight permutation, the key is divided into two 28-bit parts. Each part is shifted left (circular shift) one or two bits. In rounds 1, 2, 9, and 16, shifting is one bit; in the other rounds, it istwo bits. The two parts are then combined to form a 56-bit part. Table 6.13 shows the number of shifts for each round.

Table 6.13 Number of bit shifts

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

Table 6.14 Key-compression table

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Examples

Before analyzing DES, let us look at some examples to see the how encryption and decryption change the value of bits in each round.

Example 6.5 We choose a random plaintext block and a random key, and determine what the ciphertext block would be (all in hexadecimal):

Plaintext: 123456ABCD132536 Key: AABB09182736CCDD

Cipher Text: C0B7A8D05F3A829C

Let us show the result of each round and the text created before and after the rounds. Table 6.15 first shows the result of steps before starting the round.

Table 6.15 Trace of data for Example 6.5

Plaintext: 123456ABCD132536			
*After initial permutation:*14A7D67818CA18AD After splitting: L_0=14A7D678 R_0=18CA18AD			
Round	*Left*	*Right*	*Round Key*
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3
Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC
Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C			*(after final permutation)*

The plaintext goes through the initial permutation to create completely different 64 bits (16hexadecimal digit). After this step, the text is split into two halves, which we call L_0 and R_0. The table shows the result of 16 rounds that involve mixing and swapping (except for the last round). The result sof the last rounds (L_{16} and R_{16}) are combined. Finally the text goes through final permutation to create the ciphertext.

Some points are worth mentioning here. First, the right section out of each round is the same as the leftsection out of the next round. The reason is that the right section goes through

the mixer without change,but the swapper moves it to the left section. For example, R_1 passes through the mixer of the second roundwithout change, but then it becomes L_2 because of the swapper. The interesting point is that we do not havea swapper at the last round. That is why R_{15} becomes R_{16} instead of becoming L_{16}.

Example 6.6 Let us see how Bob, at the destination, can decipher the ciphertext received from Alice using the same key. We have shown only a few rounds to save space. Table 6.16 shows some interesting points. First, the round keys should be used in the reverse order. Compare Table 6.15 and Table 6.16. The round key for round 1 is the same as the round key for round 16. The values of L0 andR0 during decryption are the same as the values of L16 and R16 during encryption. This is the same with other rounds. This proves not only that the cipher and the reverse cipher are inverses of each other in the whole, but also that each round in the cipher has a corresponding reverse round in the reverse cipher.
The result proves that the initial and final permutation steps are also inverses of each other.
Table 6.16 Trace of data for Example 6.6

Ciphertext: C0B7A8D05F3A829C			
After initial permutation: 19BA9212CF26B472 After splitting: L_0=19BA9212 R_0=CF26B472			
Round	*Left*	*Right*	*Round Key*
Round 1	CF26B472	BD2DD2AB	181C5D75C66D
Round 2	BD2DD2AB	387CCDAA	3330C5D9A36D
. . .	. . .	. . .	. . .
Round 15	5A78E394	18CA18AD	4568581ABCCE
Round 16	14A7D678	18CA18AD	194CD072DE8C
After combination: 14A7D67818CA18AD			
Plaintext:123456ABCD132536		(after final permutation)	

DES ANALYSIS
Critics have used a strong magnifier to analyze DES. Tests have been done to measure the strength of some desired properties in a block cipher. The elements of DES have gone through scrutinies to see if they have met the established criteria. We discuss some of these in this section.
Properties
Two desired properties of a block cipher are the **avalanche effect and the completeness**.
Avalanche Effect:Avalanche effect means a small change in the plaintext (or key) should create a significant change in the ciphertext. DES has been proved to be strong with regard to this property.

Example 6.7 To check the avalanche effect in DES, let us encrypt two plaintext blocks (with thesame key) that differ only in one bit and observe the differences in the number of bits in each round.
Plaintext: 0000000000000000 Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1
Plaintext: 0000000000000001 Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3
Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits.
This means that changing approximately 1.5 percent of the plaintext creates a change of approximately45 percent in the ciphertext. Table 6.17 shows the change in each round. It shows that significantchanges occur as early as the third round.
Table 6.17 Number of bit differences for Example 6.7

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

Completeness Effect:Completeness effect means that each bit of the ciphertext needs to dependon many bits on the plaintext. The diffusion and confusion produced by D-boxes and S-boxes in DES,show a very strong completeness effect.

Design Criteria

The design of DES was revealed by IBM in 1994. Many tests on DES have proved that it satisfies some of the required criteria as claimed. Discuss some of these design issues.

S-Boxes We have discussed the general design criteria for S-boxes in Chapter 5; we only discuss the criteria selected for DES here. The design provides confusion and diffusion of bits from each round to the next. According to this revelation and some research, we can mention several properties ofS-boxes.

1. The entries of each row are permutations of values between 0 and 15.
2. S-boxes are nonlinear. In other words, the output is not an affine transformation of the input.
3. If we change a single bit in the input, two or more bits will be changed in the output.
4. If two inputs to an S-box differ only in two middle bits (bits 3 and 4), the output must differ in at least two bits. In other words, S(x) and S(x $\oplus$001100) must differ in at least two bits where x is the input and S(x) is the output.
5. If two inputs to an S-box differ in the first two bits (bits 1 and 2) and are the same in the last two bits (5 and 6), the two outputs must be different. In other words, we need to have the following relation S(x) $\neq$ S(x $\oplus$11bc00), in which b and c are arbitrary bits.
6. There are only 32 6-bit input-word pairs (xi and xj), in which xi $\oplus$xj$\neq$ (000000)2. These 32 input pairs create 32 4-bit output-word pairs. If we create the difference between the 32 output pairs,d = yi$\oplus$yj, no more than 8 of these d's should be the same.
7. A criterion similar to # 6 is applied to three S-boxes.
8. In any S-box, if a single input bit is held constant (0 or 1) and the other bits are changed randomly, the differences between the number of 0s and 1s are minimized.

D-Boxes

Between two rows of S-boxes (in two subsequent rounds), there are one straight D-box (32 to 32)and one expansion D-box (32 to 48). These two D-boxes together provide diffusion of bits.

The following criteria were implemented in the design ofD-boxes to achieve this goal:

1. Each S-box input comes from the output of a different S-box (in the previous round).
2. No input to a given S-box comes from the output from the same box (in the previous round).
3. The four outputs from each S-box go to six different S-boxes (in the next round).
4. No two output bits from an S-box go to the same S-box (in the next round).
5. If we number the eight S-boxes, S_1, S_2, …, S_8,
   a. An output of S_{j-2} goes to one of the first two bits of S_j(in the next round).
   b. An output bit from S_{j-1} goes to one of the last two bits of S_j(in the next round).
   c. An output of S_{j+1} goes to one of the two middle bits of S_j(in the next round).
6. For each S-box, the two output bits go to the first or last two bits of an S-box in the next round. The other two output bits go to the middle bits of an S-box in the next round.
7. If an output bit from $S_j$goes to one of the middle bits in S_k(in the next round), then an outputbit from $S_k$cannot go to the middle bit of S_j. If we let j = k, this implies that none of the middlebits of an S-box can go to one of the middle bits of the same S-box in the next round.

Number of Rounds DES uses sixteen rounds of Feistel ciphers. It has been proved that after eight rounds, each ciphertext is a function of every plaintext bit and every key bit; the

ciphertext is thoroughlya random function of plaintext and ciphertext. Therefore, it looks like eight rounds should be enough.
However, experiments have found that DES versions with less than sixteen rounds are even more vulnerableto known-plaintext attacks than brute-force attack, which justifies the use of sixteen rounds bythe designers of DES.

DES Weaknesses
During the last few years critics have found some weaknesses in DES.
Weaknesses in Cipher Design
Some weaknesses that have been found in the design of the cipher.
S-boxes:At least three weaknesses are mentioned in the literature for S-boxes.
1. In S-box 4, the last three output bits can be derived in the same way as the first output bit bycomplementing some of the input bits.
2. Two specifically chosen inputs to an S-box array can create the same output.
3. It is possible to obtain the same output in a single round by changing bits in only three neighboringS-boxes.

D-boxes: One mystery and one weakness were found in the design of D-boxes:
1. It is not clear why the designers of DES used the initial and final permutations; these have no security benefits.
2. In the expansion permutation (inside the function), the first and fourth bits of every 4-bit seriesare repeated.

Weakness in the Cipher Key
Several weaknesses have been found in the cipher key.
Key Size Critics believe that the most serious weakness of DES is in its key size (56 bits). To do abrute-force attack on a given ciphertext block, the adversary needs to check 256 keys.
a. With available technology, it is possible to check one million keys per second. This means that we need more than two thousand years to do brute-force attacks on DES using only a computer with one processor.
b. If we can make a computer with one million chips (parallel processing), then we can test the whole key domain in approximately 20 hours. When DES was introduced, the cost of such a computer was over several million dollars, but the cost has dropped rapidly. A special computer was built in 1998 that found the key in 112 hours.
c. Computer networks can simulate parallel processing. In 1977 a team of researchers used 3500computers attached to the Internet to find a key challenged by RSA Laboratories in 120 days.
The key domain was divided among all of these computers, and each computer was responsible to check the part of the domain.
d. If 3500 networked computers can find the key in 120 days, a secret society with 42,000 members can find the key in 10 days. The above discussion shows that DES with a cipher key of 56 bits is not safe enough to be used comfortably. We will see later in the chapter that one solution is to use triple DES (3DES) with two keys(112 bits) or triple DES with three keys (168 bits).

Weak Keys Four out of 256 possible keys are called **weak keys.** A weak key is the one that, after parity drop operation (using Table 6.12), consists either of all 0s, all 1s, or half 0s and half 1s. These keysare shown in Table 6.18.
Table 6.18 Weak keys

Keys before parities drop (64 bits)	*Actual key (56 bits)*
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

The round keys created from any of these weak keys are the same and have the same pattern as the cipher key. For example, the sixteen round keys created from the fi rst key is all made of 0s; the one from the second is made of half 0s and half 1s. The reason is that the key-generation algorithm first divides the cipher key into two halves. Shifting or permutation of a block does not change the block if it is made of all 0s or all 1s.

What is the disadvantage of using a weak key? If we encrypt a block with a weak key and subsequently encrypt the result with the same weak key, we get the original block. The process creates the same original block if we decrypt the block twice. In other words, each weak key is the inverse of itself Ek(Ek(P)) = P, as shown in Fig. 6.11.

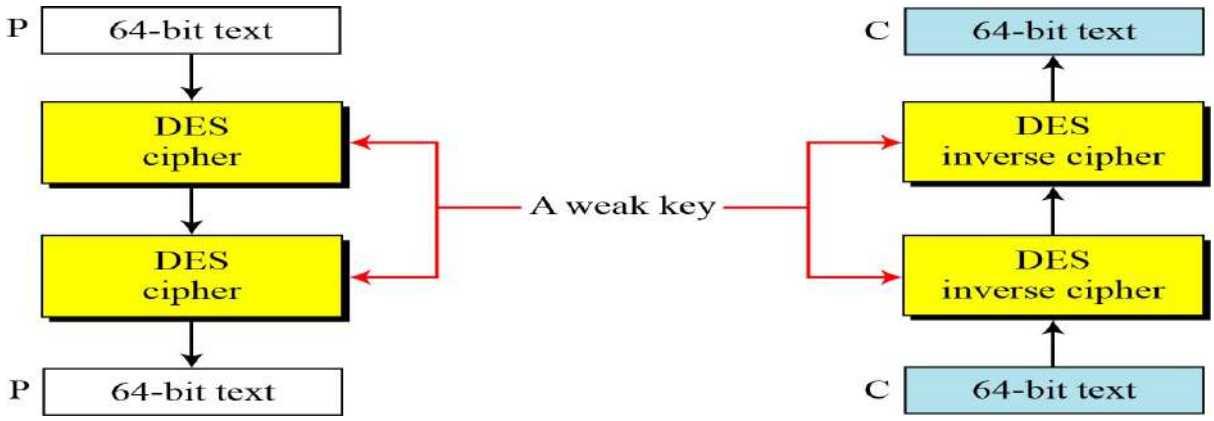


Fig. 6.11 Double encryption and decryption with a weak key

Weak keys should be avoided because the adversary can easily try them on the intercepted ciphertext.If after two decryptions the result is the same, the adversary has found the key.

Example 6.8 Let us try the fi rst weak key in Table 6.18 to encrypt a block two times. After twoencryptions with the same key the original plaintext block is created. Note that we have used theencryption algorithm two times, not one encryption followed by another decryption.

Key: 0x0101010101010101

Plaintext: 0x1234567887654321Ciphertext: 0x814FE938589154F7

Key: 0x0101010101010101

Plaintext: 0x814FE938589154F7Ciphertext: 0x1234567887654321

Semi-weak Keys:There are six key pairs that are called **semi-weak keys.** These six pairs are shownin Table 6.19 (64-bit format before dropping the parity bits).

Table 6.19 Semi-weak keys

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

A semi-weak key creates only two different round keys and each of them is repeated eight times. Inaddition, the round keys created from each pair are the same with different orders. To show the idea, wehave created the round keys from the first pairs as shown below:

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

As the list shows, there are eight equal round keys in each semi-weak key. In addition, round key 1 in the first set is the same as round key 16 in the second; round key 2 in the first is the same as roundkey 15 in the second; and so on. This means that the keys are inverses of each other $E_{k2}(E_{k1}(P)) = P$, asshown in Fig. 6.12.

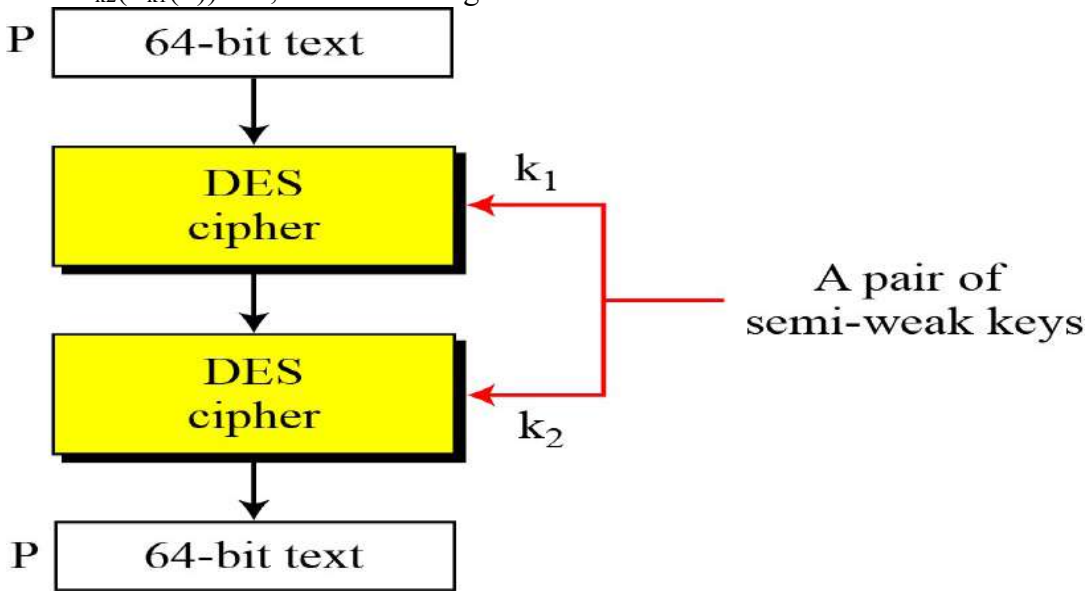


Fig. 6.12 A pair of semi-weak keys in encryption and decryption

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

Example 6.9 What is the probability of randomly selecting a weak, a semi-weak, or a possible weakkey?

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

Key Complement In the key domain (2^{56}), definitely half of the keys are complement of the otherhalf.

A **key complement** can be made by inverting (changing 0 to 1 or 1 to 0) each bit in the key. Doesa key complement simplify the job of the cryptanalysis? It happens that it does. Eve can use only half ofthe possible keys (2^{55}) to perform brute-force attack. This is because

$$C = E(K, P) \rightarrow \overline{C} = E(\overline{K}, \overline{P})$$

In other words, if we encrypt the complement of plaintext with the complement of the key, we get thecomplement of the ciphertext. Eve does not have to test all 2^{56} possible keys, she can test only half ofthem and then complement the result.

Example 6.10 Let us test the claim about the complement keys. We have used an arbitrary key and plaintext to find the corresponding ciphertext. If we have the key complement and the plaintext, we canobtain the complement of the previous ciphertext (Table 6.20).

Table 6.20 Results for Example 6.10

	Original	*Complement*
Key	1234123412341234	EDCBEDCBEDCBEDCB
Plaintext	12345678ABCDEF12	EDCBA987543210ED
Ciphertext	E112BE1DEFC7A367	1EED41E210385C98

Key Clustering: Key clustering refers to the situation in which two or more different keys can createthe same ciphertext from the same plaintext. Obviously, each pair of the semi-weak keys is a key cluster.However, no more clusters have been found for the DES. Future research may reveal some more.

SECURITY OF DES

DES, as the first important block cipher, has gone through much scrutiny. Among the attempted attacks,three are of interest: brute-force, differential cryptanalysis, and linear cryptanalysis.

1 Brute-Force Attack

We have discussed the weakness of short cipher key in DES. Combining this weakness with the keycomplement weakness, it is clear that DES can be broken using 2^{55} encryptions. However, today mostapplications use either 3DES with two keys (key size of 112) or 3DES with three keys (key size of 168).These two multiple-DES versions make DES resistant to brute-force attacks.

2 Differential Cryptanalysis

We discussed the technique of differential cryptanalysis on modern block ciphers in Chapter 5. DESis not immune to that kind of attack. However, it has been revealed that the designers of DES alreadyknew about this type of attack and designed S-boxes and chose 16 as the number of rounds to make DES specifically resistant to this type of attack. Today, it has been shown that DES can be brokenusing differential cryptanalysis if we have 247 chosen plaintexts or 255 known plaintexts. Although thislooks more efficient than a brute-force attack, finding 247 chosen plaintexts or 255 know plaintexts isimpractical. Therefore, we can say that DES is resistant to differential cryptanalysis. It has also beenshown that increasing the number of rounds to 20 require more than 264 chosen plaintexts for this attack,which is impossible because the possible number of plaintext blocks in DES is only 264.

3 Linear Cryptanalysis

We discussed the technique of linear cryptanalysis on modern block ciphers in Chapter 5. Linearcryptanalysis is newer than differential cryptanalysis. DES is more vulnerable to linear cryptanalysisthan to differential cryptanalysis, probably because this type of attack was not known to the designers ofDES. S-boxes are not very resistant to linear cryptanalysis. It has been shown that DES can be brokenusing 243 pairs of known plaintexts. However, from the practical point of view, finding so many pairsis very unlikely.

MULTIPLE DES—CONVENTIONAL ENCRYPTION ALGORITHMS

If a block cipher has a key size, which is small in context to the present day computation power, then a natural way out may be to perform multiple encryptions by the block cipher. As an example, consider the DES algorithm which has a key size of 56 bits, which is short in context to the modern computation capability. The threat is that such a key value can be

evaluated by brute force key search. Hence two DES applications give what is known as 2-DES.

Double DES (2-DES) and Meet in the Middle Attack

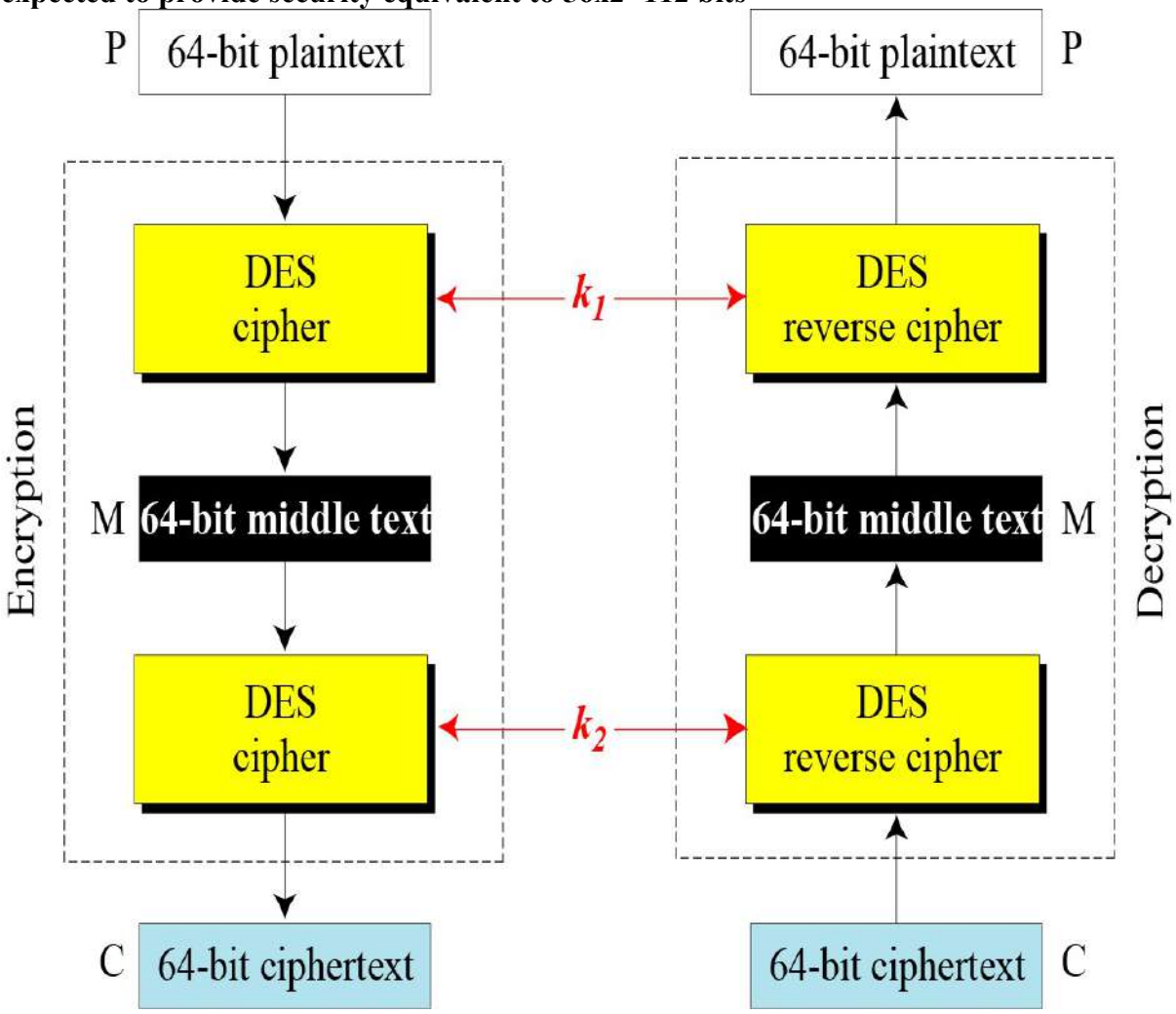
Consider a message m, which is to be encrypted. The corresponding block cipher for one application ofthe DES applications is represented by Ek, where k is the corresponding DES key. The output of 2-DESis c = $E_{k2}(E_{k1}(m))$. To decrypt similarly, m = $D_{k1}(D_{k2}(c))$. This cipher, 2-DES should offer additional security, equivalent to both k1 and k2. The cipher 2-DES obtained by the repeated application of DESis called, 2 – DES = DES xDES. This is called a product cipher obtained by the composition of twociphers. Such an idea can similarly be extended to multiple ciphers.

It may be noted that such a product on the DES cipher is expected to provide additional security, because

DES does not form a group under the composition operation. That is the composition (application) oftwo ciphers with two different keys cannot be obtained by a single application of DES with a key. Thus 2-DES is expected to provide security equivalent to 56x2=112 bits. However it can be shown that sucha cipher can be attacked by an attack method which is called Meet-in-the-Middle attack.

Meet-in-the-Middle Attack: Double DES

- **However, using a known-plaintext attack called meet-in-the-middle attack proves that double DES improves this vulnerability slightly (to 2^{57} tests), but not tremendously (to 2^{112}).**
- **2-DES is expected to provide security equivalent to 56x2=112 bits2-DES is expected to provide security equivalent to 56x2=112 bits**
-



Meet-in-the-Middle (MIM) Attack and 3-DES

Consider the cipher 2-DES as defined above. The plaintext and the ciphertext of the cipher is $P=\{0,1\}^m$.The key space of DES is $K=\{0,1\}^n$, the key size of the product cipher is expected to be K_1xK_2, wherethe key is represented as the ordered pair (k1,k2), where k1 belongs to K1 and k_2 belongs to K_2.

The attacker obtainslpairs of plaintexts and ciphertexts: $(p_1,c_1),\ldots,(p_l,c_1)$. The key is say (K_1,K_2) butunknown to the attacker (obviously, else why will he/she be an attacker).

It is easy to prove that for all $1 \leq i \leq l,\ DES_{K_1}(p_i) = \mathrm{DES}^{-1}_{K_2}(c)$. . There are in total 2^{2n} keys. Theprobability of a key satisfying this equation for a particular value of i is 2-m, as that is the block size ofthe cipher. Since all the ivalues of the plaintext, ciphertext pairs are independent, the probability of akey satisfying the above equation for all the l values of i, is 2^{-ml}.

Triple DES with Three Keys

The possibility of known-plaintext attacks on triple DES with two keys has enticed some applications to use triple DES with three keys. Triple DES with three keys is used by many applications such as PGP

Pretty Good Privacy (**PGP**) is an encryption program that provides cryptographic privacy and authentication for data communication.