DATA ENCRYPTION STANDARD

All worked examples in the correct order of the DES algorithm

EXAMPLE USED THROUGHOUT

Plaintext: 0123456789ABCDEF **Key: 133457799BBCDFF1**

THE ENCRYPTION SEQUENCE

01 Initial permutation	**02** Split left / right	**03** Round-key setup	**04** Expansion (E)	**05** XOR with key
06 S-box substitution	**07** P permutation	**08** Update L and R	**09** 16 rounds + swap	**10** Inverse IP

How to use this guide

Blue/orange/green panels show original bits, operations, and results.

Miniature examples teach the rule; full-size examples use real DES values.

INITIAL PERMUTATION (IP)

A clear 8-bit example: rearrange positions, not values

DES ENCRYPTION

IDEA: IP copies bits into a new order using a fixed list of original positions.

1 Original 8 bits

Position	1	2	3	4	5	6	7	8
	1	0	1	1	0	0	1	0

2 Given permutation rule

2	4	6	8	1	3	5	7

Read this rule from left to right.

3 Move each bit into its new position

NEW POSITION	1	2	3	4	5	6	7	8
TAKE BIT FROM	2	4	6	8	1	3	5	7
BIT COPIED	0	1	0	0	1	1	0	1

Example: New bit 1 = original bit 2 = 0 | New bit 2 = original bit 4 = 1

FINAL OUTPUT **IP(10110010) = 01001101**

ACTUAL DES: INITIAL PERMUTATION AND SPLIT

Connect the 8-bit teaching example to the real 64-bit input

DES | ENCRYPTION

REAL 64-BIT PLAINTEXT

0123456789ABCDEF

After applying the standard 64-position IP table

CC00CCFFF0AAF0AA

↓

LEFT HALF / L_0

CC00CCFF

RIGHT HALF / R_0

F0AAF0AA

1 **Both halves are 32 bits long.**

R_0 is sent into the round function F. L_0 is retained for the half-update XOR.

KEY IDEA IP rearranges bits; splitting does not change or discard any bit.

LEFT AND RIGHT HALVES: SIMPLE EXAMPLE

See how each half moves during one DES-style Feistel round

DES | ENCRYPTION

IDEA: In each round: new left = old right; new right = old left XOR the round-function output.

1 Start with 8 bits (a miniature example to learn the rule)

1	2	3	4	5	6	7	8
1	0	1	1	0	0	1	0

LEFT (L_0) = 1011 **RIGHT (R_0) = 0010**

2 Use the two Feistel round rules

RULE A Copy the old right half

$$L_1 = R_0 = 0010$$

RULE B XOR the old left half with F

$$R_1 = L_0 \text{ XOR } F(R_0, K_1)$$

GIVEN (to keep this example easy):

$F(R_0, K_1) = 0110$ $R_1 = 1011 \text{ XOR } 0110 = 1101$

AFTER ROUND 1 $L_1 = 0010$ | $R_1 = 1101$

ROUND KEY: WHERE DOES K_1 COME FROM?

The DES key schedule supplies a fresh 48-bit subkey for each of the 16 rounds

DES | ENCRYPTION

STARTING SECRET KEY **133457799BBCDFF1**

1. **64 key bits** — Eight parity bits exist
2. → **PC-1: 56 bits** — Select and reorder
3. → **C_0 + D_0** — Two 28-bit halves
4. → **Left rotation** — Shift halves for round 1
5. → **PC-2: 48 bits** — Select / permute bits

ROUND 1 SUBKEY

K_1 = 1B02EFFC7072

This 48-bit key is XORed with the expanded 48-bit right half.

Why do we need this step?

Each round has a different key K_i; the full algorithm runs 16 rounds.

The expansion and XOR examples that follow use K_1 shown above.

EXPANSION BOX (E-BOX): SIMPLE EXAMPLE

First understand the idea using just 8 input bits

DES | ENCRYPTION

IDEA: An E-box rearranges AND repeats selected bits to produce a longer bit sequence.

1 Start with an 8-bit input and number its positions

1	2	3	4	5	6	7	8
1	0	1	1	0	0	1	0

Input bits: 1 0 1 1 | 0 0 1 0

2 Use a small expansion rule: two groups of 4 bits become two groups of 6

GROUP 1 Previous | 4 original | Next

Take positions:	8	1	2	3	4	5
Their bit values:	0	1	0	1	1	0

GROUP 2 Previous | 4 original | Next

Take positions:	4	5	6	7	8	1
Their bit values:	1	0	0	1	0	1

EXPANDED RESULT

Two groups of 6 bits = 12 output bits

010110 | 100101

010110100101 (12 bits)

DES EXPANSION: START WITH F0AAF0AA

A visual explanation of how 32 input bits become 48 output bits

DES | ENCRYPTION

MAIN IDEA: Every set of 4 bits becomes 6 by borrowing ONE neighboring bit on each side.

1 Convert each hexadecimal digit into four binary bits

Given hexadecimal input: **F0AAF0AA**

Hex digit	F	0	A	A	F	0	A	A
Four bits	1111	0000	1010	1010	1111	0000	1010	1010

2 Divide the 32 bits into eight 4-bit groups

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8
1111	0000	1010	1010	1111	0000	1010	1010

CHECK: 8 groups × 4 bits = 32 original bits.

Next: take the last bit of the previous group and the first bit of the next group.

HOW ONE 4-BIT GROUP BECOMES 6 BITS

Join (do not add): previous last bit | current four bits | next first bit

DES | ENCRYPTION

RULE: Read the groups in a circle: before Group 1 comes Group 8; after Group 8 comes Group 1.

EXAMPLE 1 | Expand Group 1 = 1111

Wrap-around: the previous group of Group 1 is Group 8 (1010); the next is Group 2 (0000).

PREVIOUS: Group 8	CURRENT: Group 1	NEXT: Group 2
0	1111	0

0 | 1111 | 0 -> 011110

EXAMPLE 2 | Expand Group 2 = 0000

For Group 2, take Group 1's last bit (1) and Group 3's first bit (1).

PREVIOUS: Group 1	CURRENT: Group 2	NEXT: Group 3
1	0000	1

1 | 0000 | 1 -> 100001

COMPLETE EXPANSION: ALL EIGHT GROUPS

Repeat the same neighbor-bit rule for every group, then combine the answers

DES | ENCRYPTION

FORMULA: Expanded group = Previous last bit | Original 4 bits | Next first bit

GROUP	LEFT BIT	ORIGINAL 4	RIGHT BIT	EXPANDED 6 BITS
1	0	1111	0	**011110**
2	1	0000	1	**100001**
3	0	1010	1	**010101**
4	0	1010	1	**010101**
5	0	1111	0	**011110**
6	1	0000	1	**100001**
7	0	1010	1	**010101**
8	0	1010	1	**010101**

COMBINE ALL EIGHT 6-BIT GROUPS:

011110 100001 010101 010101 011110 100001 010101 010101

48 binary bits = 0111 1010 0001 0101 0101 0101 0111 1010 0001 0101 0101 0101

HEX OUTPUT: **7A15557A1555**

REAL DES E-BOX: 32 BITS TO 48 BITS

The same concept: copy selected boundary bits and keep their prescribed order

DES | ENCRYPTION

RULE: DES uses a fixed expansion table: eight 4-bit sections each become 6 bits.

32-BIT INPUT (right half R) **F0AAF0AA**

Binary (four groups of 8):

11110000	10101010	11110000	10101010

1 Read six positions from each row, then copy the bits at those positions

GROUP	SOURCE BIT POSITIONS	BITS TAKEN FROM INPUT	6-BIT OUTPUT
E1	32 1 2 3 4 5	0 1 1 1 1 0	011110
E2	4 5 6 7 8 9	1 0 0 0 0 1	100001
E3	8 9 10 11 12 13	0 1 0 1 0 1	010101
E4	12 13 14 15 16 17	0 1 0 1 0 1	010101
E5	16 17 18 19 20 21	0 1 1 1 1 0	011110
E6	20 21 22 23 24 25	1 0 0 0 0 1	100001
E7	24 25 26 27 28 29	0 1 0 1 0 1	010101
E8	28 29 30 31 32 1	0 1 0 1 0 1	010101

48-BIT EXPANDED OUTPUT

Some input bits appear twice; no new information is created.

011110 100001 010101 010101 011110 100001 010101 010101

Hexadecimal: 7A15557A1555

XOR OPERATION: SIMPLE SOLVED EXAMPLE

Understand the four XOR rules, then compare two 8-bit strings

DES | ENCRYPTION

IDEA: XOR compares TWO bits at the same position: same = 0, different = 1.

1 THE FOUR XOR RULES

BIT A	BIT B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

REMEMBER THIS ONE RULE

DIFFERENT bits give 1

0 XOR 1 = 1 1 XOR 0 = 1

SAME bits give 0

0 XOR 0 = 0 1 XOR 1 = 0

2 Compare two 8-bit values position by position

BIT POSITION	1	2	3	4	5	6	7	8
INPUT A	1	0	1	1	0	0	1	0
INPUT B	0	1	1	0	1	0	0	1
A XOR B	1	1	0	1	1	0	1	1

FINAL 8-BIT RESULT

10110010 XOR 01101001 = 11011011

XOR IN DES: A REAL ROUND EXAMPLE

XOR the 48-bit expanded right half with the 48-bit round key

DES | ENCRYPTION

WHY? DES combines expanded data with its round subkey using bit-by-bit XOR.

1 Take the two equal-length inputs (48 bits each)

EXPANDED RIGHT HALF

7A15557A1555

XOR

ROUND KEY K1

1B02EFFC7072

2 Perform XOR one byte (8 bits) at a time

BYTE	DATA	KEY	XOR	BINARY WORKING (DATA XOR KEY = RESULT)
1	7A	1B	61	01111010 XOR 00011011 = 01100001
2	15	02	17	00010101 XOR 00000010 = 00010111
3	55	EF	BA	01010101 XOR 11101111 = 10111010
4	7A	FC	86	01111010 XOR 11111100 = 10000110
5	15	70	65	00010101 XOR 01110000 = 01100101
6	55	72	27	01010101 XOR 01110010 = 00100111

48-BIT XOR RESULT

7A15557A1555 XOR 1B02EFFC7072 = 6117BA866527

SUBSTITUTION BOX (S-BOX): ONE LOOKUP

The easiest way to understand how a DES S-box works

DES | ENCRYPTION

IDEA: An S-box uses 6 input bits to look up a number, then produces 4 output bits.

1 Read the six input bits

0	1	1	0	0	0
1	2	3	4	5	6

2 Find row and column

FIRST + LAST = 00 = row 0

MIDDLE FOUR = 1100 = column 12

3 Look up row 0, column 12 in the real DES S1 table

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

RESULT

S1[0, 12] = 5

Decimal 5 = Binary 0101

6 input bits -> 4 output bits

ALL EIGHT S-BOXES: A COMPLETE EXAMPLE

Apply the same row-column rule eight times; each box uses its own table

DES | ENCRYPTION

IDEA: 48 bits = eight groups of 6. Each S-box gives 4 bits. Total output = 32 bits.

48-BIT INPUT (hexadecimal) **6117BA866527**

Binary groups: **011000** **010001** **011110** **111010** **100001** **100110** **010100** **100111**

1 Use the outer bits for the row and the middle four bits for the column

S-BOX	6-BIT INPUT	ROW (outer)	COLUMN (middle)	S-BOX VALUE	4-BIT OUTPUT
S1	011000	00 = 0	1100 = 12	**5**	**0101**
S2	010001	01 = 1	1000 = 8	**12**	**1100**
S3	011110	00 = 0	1111 = 15	**8**	**1000**
S4	111010	10 = 2	1101 = 13	**2**	**0010**
S5	100001	11 = 3	0000 = 0	**11**	**1011**
S6	100110	10 = 2	0011 = 3	**5**	**0101**
S7	010100	00 = 0	1010 = 10	**9**	**1001**
S8	100111	11 = 3	0011 = 3	**7**	**0111**

FINAL OUTPUT

Eight 4-bit answers joined together

0101 1100 1000 0010 1011 0101 1001 0111

32 bits = 5C82B597 (hex)

P-PERMUTATION: REARRANGE 32 BITS

After all eight S-boxes, DES permutes the 32-bit result without changing its length

DES | ENCRYPTION

OUTPUT FROM THE EIGHT S-BOXES

5C82B597

P permutation table (each entry identifies an input bit position)

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

FIRST OUTPUT BYTE

bits at positions 16, 7, 20, 21, 29, 12, 28, 17
= 00100011 (binary)

RESULT OF ROUND FUNCTION F(R_0, K_1)

P(5C82B597) = 234AA9BB

ACTUAL DES: MANAGING THE TWO 32-BIT HALVES

Use the same DES values from the IP, E-box, S-box and XOR lectures

START: After the Initial Permutation, split the 64-bit result into two 32-bit halves.

1 Split the permuted 64-bit input

IP(plaintext) = CC00CCFFF0AAF0AA

L_0 = CC00CCFF

R_0 = F0AAF0AA

2 Calculate the round function using the right half

R_0 = F0AAF0AA → $F(R_0, K_1)$ → 234AA9BB

3 Apply the two half-update rules

NEW LEFT = OLD RIGHT

L_1 = F0AAF0AA

NEW RIGHT = OLD LEFT XOR F

CC00CCFF XOR 234AA9BB

R_1 = EF4A6544

ROUND 1 OUTPUT: $L_1 \| R_1$ = F0AAF0AAEF4A6544

REPEAT THE ROUND PROCESS 16 TIMES

Round 1 is worked out fully; rounds 2–16 reuse the same structure with their own keys

DES | ENCRYPTION

$$L_i = R_{i-1} \qquad R_i = L_{i-1} \text{ XOR } F(R_{i-1}, K_i)$$

After Round 1

L_1

F0AAF0AA

R_1

EF4A6544

Continue through Rounds 2, 3, ..., 16

AFTER ROUND 16

$$L_{16} = 43423234$$

AFTER ROUND 16

$$R_{16} = 0A4CD995$$

SWAP HALVES BEFORE FINAL IP INVERSE

$$R_{16} \,||\, L_{16} = 0A4CD99543423234$$

INVERSE INITIAL PERMUTATION (IP^{-1})

A clear 8-bit example: put each bit back into its original position

DES ENCRYPTION

IDEA: Inverse IP reverses the earlier permutation and restores the original bit order.

1 Input: bits after IP

1	2	3	4	5	6	7	8
0	1	0	0	1	1	0	1

2 Inverse permutation rule

5	1	6	2	7	3	8	4

Read this rule from left to right.

3 Move each bit back to its original position

ORIGINAL POSITION	1	2	3	4	5	6	7	8
TAKE BIT FROM IP OUTPUT	5	1	6	2	7	3	8	4
RESTORED BIT	1	0	1	1	0	0	1	0

Example: Original bit 1 = IP-output bit 5 = 1 | Original bit 2 = IP-output bit 1 = 0

FINAL OUTPUT

$$IP^{-1}(01001101) = 10110010$$

DES ENCRYPTION: FINAL CHECK

After 16 rounds, swap the halves and apply the inverse initial permutation

DES | ENCRYPTION

64-BIT BLOCK AFTER THE LAST SWAP

0A4CD99543423234

Apply the standard 64-bit inverse IP table

↓

FINAL 64-BIT CIPHERTEXT

85E813540F0AB405

CORRECT DES TEST VECTOR

Plaintext 0123456789ABCDEF **Key** 133457799BBCDFF1

Inverse IP is a final rearrangement of encrypted bits; it is not decryption.