Data Encryption Standard (DES)

Deep(ish) Dive

Cracking DES

“Before DES was adopted as a national standard, during the period the National Bureau of Standards was soliciting comments on the proposed algorithm, the creators of public key cryptography, Martin Hellman and Whitfield Diffie, registered some objections to the use of DES as an encryption algorithm.

Hellman wrote: "Whit, Diffie, and I have become concerned that the proposed data encryption standard, while probably secure against commercial assault, may be extremely vulnerable to attack by an intelligence organization" (letter to NBS, October 22, 1975).”

Cracking DES

“Diffie and Hellman then outlined a "brute force" attack on DES. (By "brute force" is meant that you try as many of the 2^56 possible keys as you have to before decrypting the ciphertext into a sensible plaintext message.) They proposed a special purpose "parallel computer using one million chips to try one million keys each" per second, and estimated the cost of such a machine at $20 million.”

Cracking DES

“Fast forward to 1998. Under the direction of John Gilmore of the EFF, a team spent $220,000 and built a machine that can go through the entire 56-bit DES key space in an average of 4.5 days. On July 17, 1998, they announced they had cracked a 56-bit key in 56 hours. The computer, called Deep Crack, uses 27 boards each containing 64 chips, and is capable of testing 90 billion keys a second.

Despite this, as recently as June 8, 1998, Robert Litt, principal associate deputy attorney general at the Department of Justice, denied it was possible for the FBI to crack DES: "Let me put the technical problem in context: It took 14,000 Pentium computers working for four months to decrypt a single message . . . . We are not just talking FBI and NSA [needing massive computing power], we are talking about every police department."

Responded cryptography expert Bruce Schneier: " . . . the FBI is either incompetent or lying, or both." Schneier went on to say: "The only solution here is to pick an algorithm with a longer key; there isn't enough silicon in the galaxy or enough time before the sun burns out to brute- force triple-DES" (Crypto-Gram, Counterpane Systems, August 15, 1998).”

Data Encryption Standard

DES is a block cipher.

DES encrypts data in blocks of size of 64 bit each - meaning 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text.

The same algorithm and key are used for encryption and decryption, with minor differences.

The key length is 56 bits.

56-Bit Key

The Initial Key consists of 64 bits.

A 56 bit key is created from the Initial Key for performing the DES process.

Before the DES begins, every 8th bit of the key is discarded to produce a 56 bit key. That is bit position 8, 16, 24, 32, 40, 48, 56 and 64 are discarded.

64 Bit Initial Key

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

DES Cryptography Concepts

DES is based on the two fundamental methods of cryptography

Substitution and Transposition

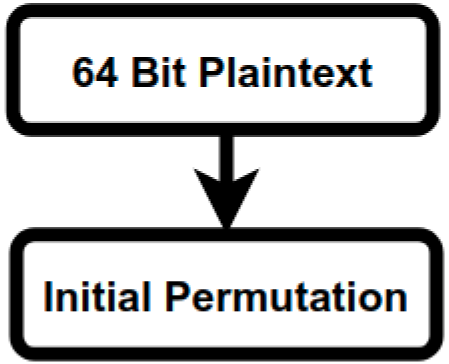
DES consists of 16 steps, each of which is called as a round.

Each round performs the steps of substitution and transposition.

DES Overview

In the first step, the 64 bit plain text block is handed over to an Initial Permutation (IP) function.

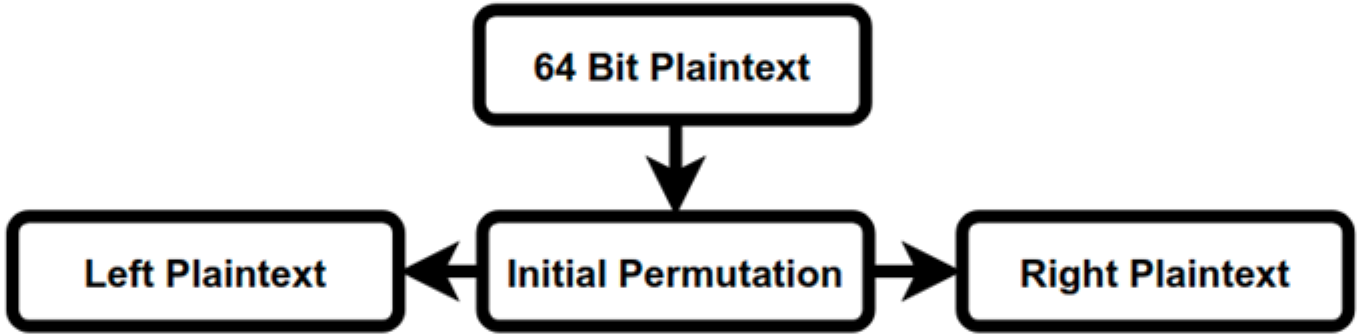
The initial permutation is performed on plain text.



DES Overview

Next the initial permutation (IP) produces two halves of the permuted block.

These halves are called - Left Plaintext (LPT) and Right Plaintext (RPT).

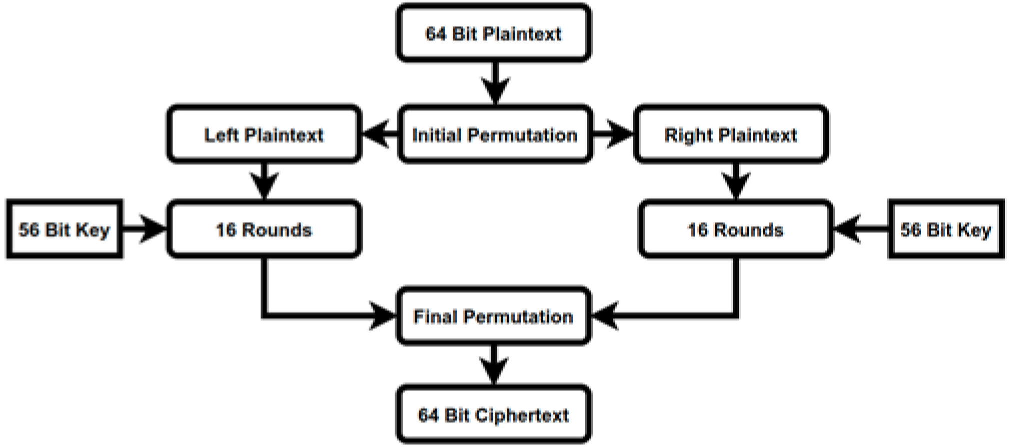


DES Overview

Now each LPT and RPT to go through 16 rounds of encryption.

In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block

The result of this process produces 64 bit Ciphertext.



Initial Permutation (IP)

The **Initial Permutation is a transposition function** that occurs only once at the beginning of the DES process.

For example, the IP **swaps** the first bit of the original Plaintext block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block etc, etc...

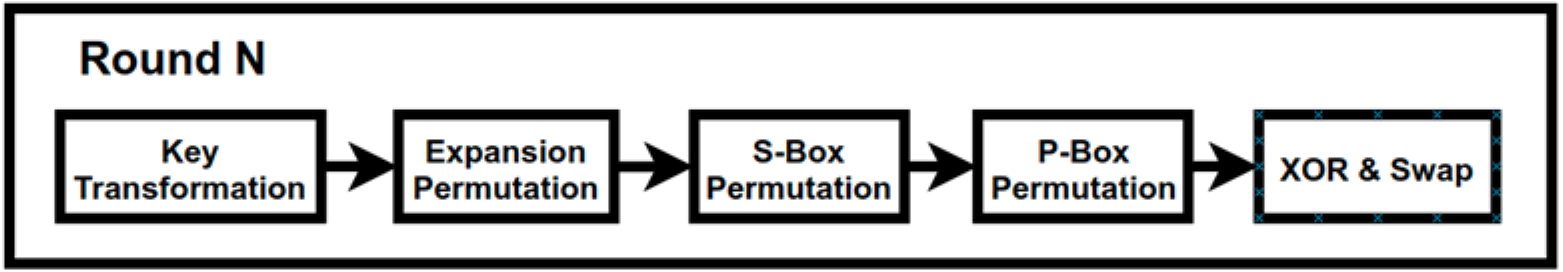
64 Bit Plaintext Block

Bit 0	58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
Bit 16	62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
Bit 32	57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
Bit 48	61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Rounds

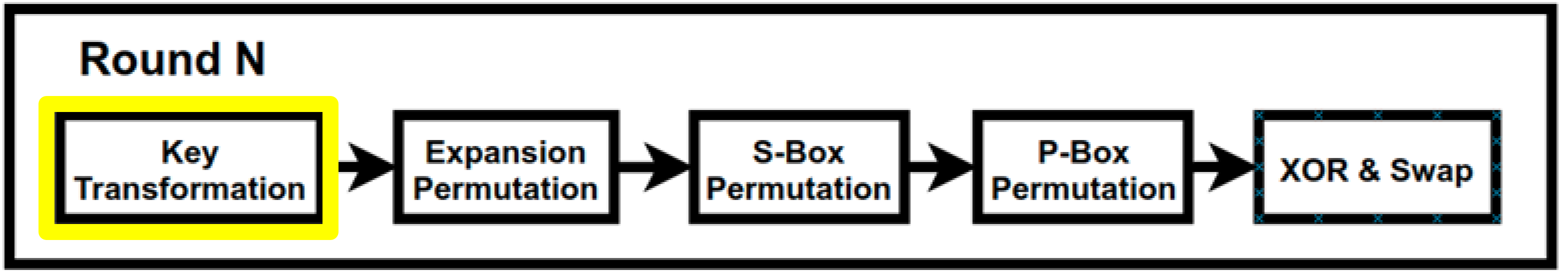
After IP is finished, the resulting 64-bit permuted text block is divided into two half-blocks.

Each half-block (LPT & RPT) consists of 32 bits, and are respectively processed through 16 rounds of cryptography.



Step 1: Key Transformation

A new **48-bit Sub Key** is generated during each round using a process called **Key Transformation**.



Step 1: Key Transformation

The 56-bit key is first divided into two halves, each of 28-bits.

These halves are circularly shifted left by one or two positions, based on the round number.

Round #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of Key Bits Shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

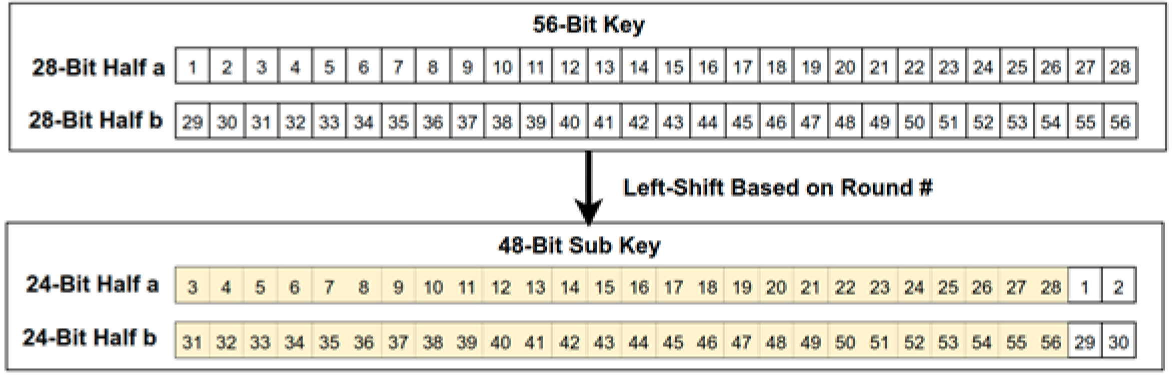
Step 1: Key Transformation

After shifting, **48 of the 56 bits are selected**.

Because transformation process involves **permutation** as well as **selection** of a 48-bit subset of the 56-bit key, this process is also sometimes called **Compression Permutation**.

Because of Compression Permutation, **a different subset of key bits is used in each round**. This makes DES difficult to break.

The two remaining 24-bit halves of the 48 bit subkey are our **Round Keys**.



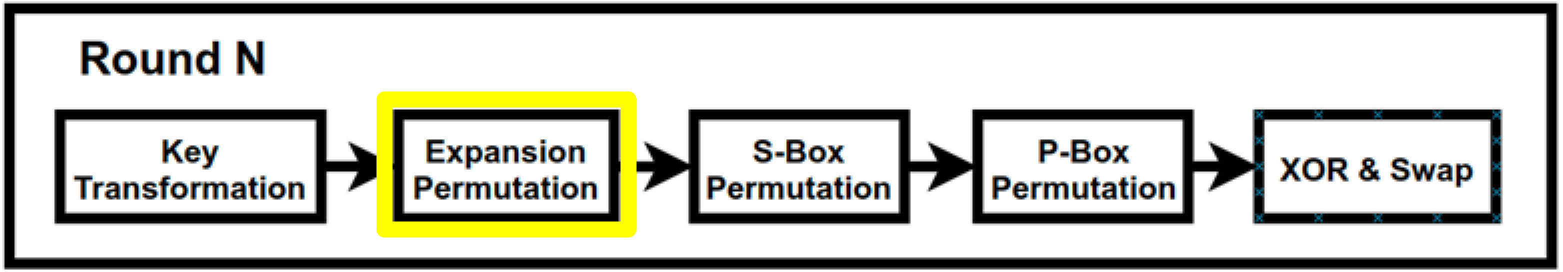
Subkey
=
Half A + Half B
(concatenated)

Step-2: Expansion Permutation

Recall that after initial permutation, we had two 32-bit Plaintext areas called Left Plaintext (LPT) and Right Plaintext (RPT).

During the Expansion Permutation, the **RPT is expanded from 32 bits to 48 bits**.

We also **permute the bits** (hence the name of the step).

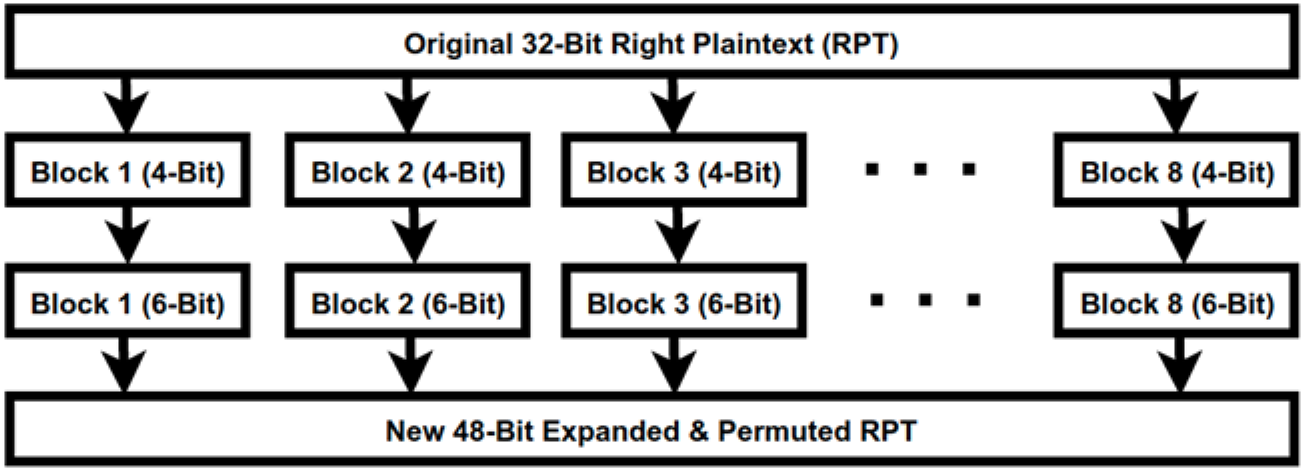


Step-2: Expansion Permutation

32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits.

Each 4-bit block is then expanded to a corresponding 6 bit block (add 2 extra bits).

2 added bits are determined by a Look-Up-Table using the 4-Bit Block value as input.

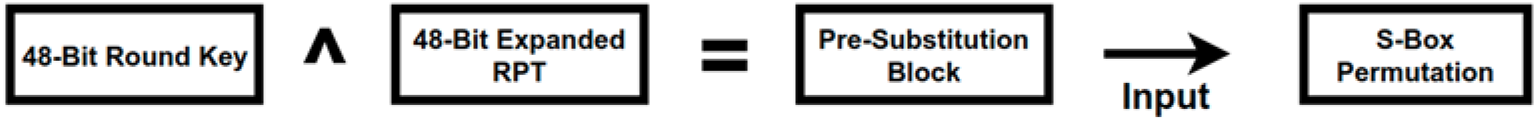


Step-2: Expansion Permutation

Now we have a 48-Bit Expanded value for the 32-Bit RPT.

Before we move on to the next step, we must XOR the 48-Bit Expanded RPT with the 48-Bit Round Key generated during Key Transformation.

This is called our **Pre-Substitution Block**.



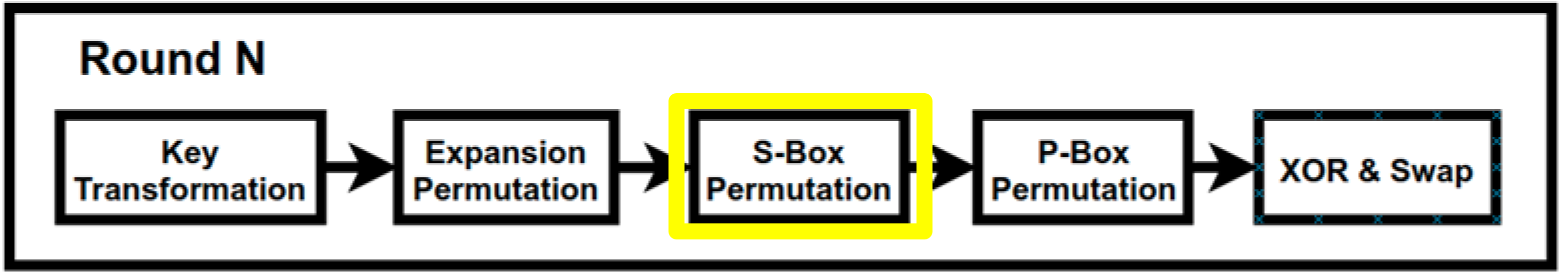
Step-3: S-Box Permutation

Substitution Boxes, or S-Boxes are used to obscure the relationship between the key and the ciphertext.

In general, an S-box takes some number of input bits, N, and transforms them into some number of output bits, M.

An m*n S-box is typically implemented as a Lookup Table of 2^m words of n bits each.

DES uses a fixed S-box table, though other algorithms may use dynamically generated S-box tables.

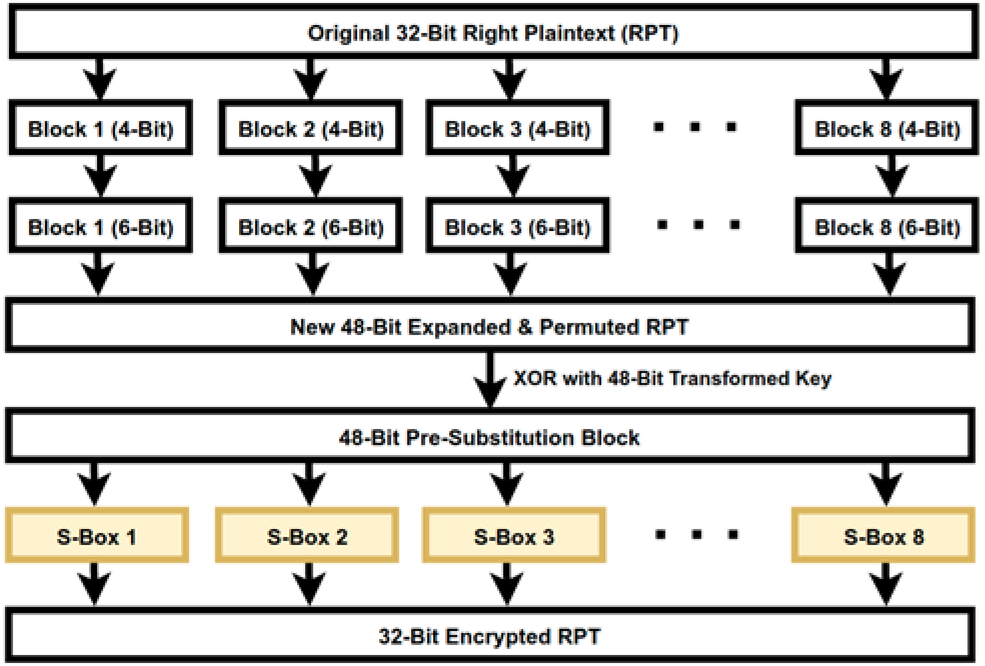


Step-3: S-Box Permutation

S-Boxes are used at this stage in the round to turn each of the 6-Bit blocks transformed 4-Bit blocks.

Each 6-Bit Block is individually substituted used an S-Box table.

After the S-Box operation we are left with a new ciphered 32-Bit vector.

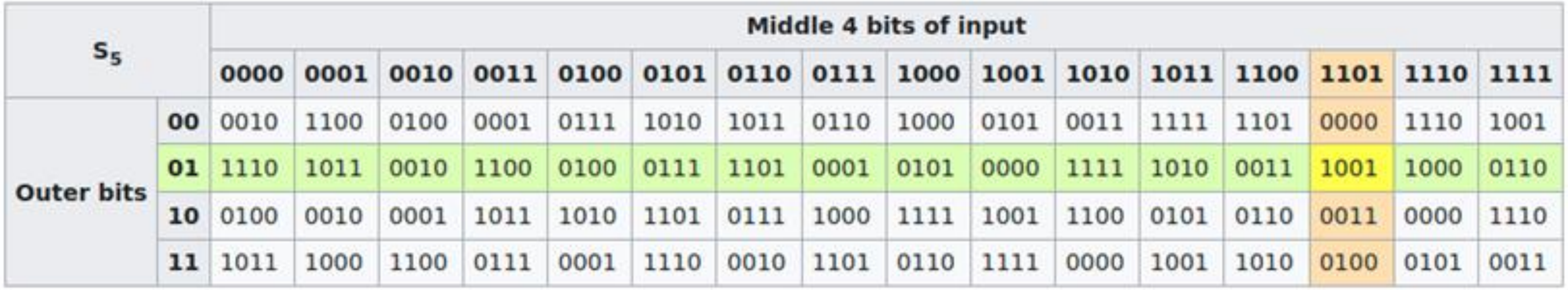


Step-3: S-Box Permutation

Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits (the first and last bits).

The column is selected using the inner four bits.

For example, an input 011011 has outer bits 01 and inner bits 1101 - the corresponding output would be 1001.



S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Step-3: S-Box Permutation

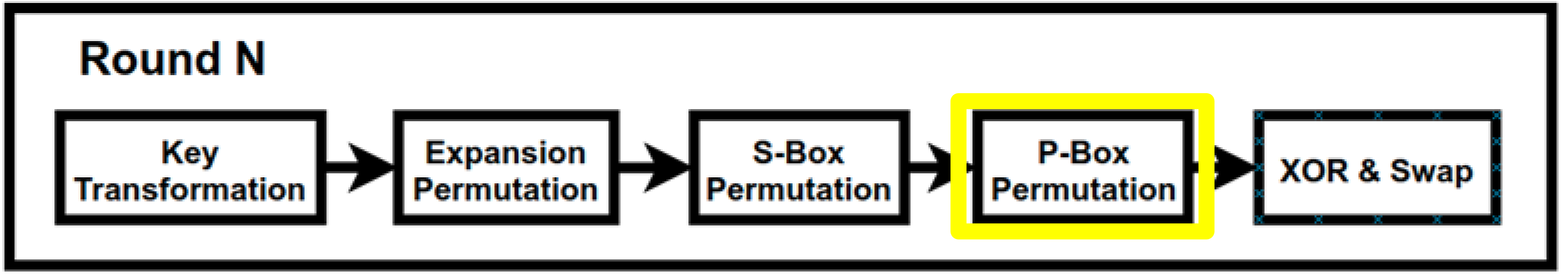
Each individual 6-to-4 bit S-Box block conversion is then concatenated together to yield a 32-bit vector.

$$S_1(B_1) \mid S_2(B_2) \mid S_3(B_3) \mid S_4(B_4) \mid S_5(B_5) \mid S_6(B_6) \mid S_7(B_7) \mid S_8(B_8)$$

Step-4: P-Box Permutation

Once we have our ‘ciphered’ 32-Bit R value, another permutation (P) phase is performed on the substituted vector.

$$f = P(S_1(B_1) \mid S_2(B_2) \mid \dots \mid S_8(B_8))$$



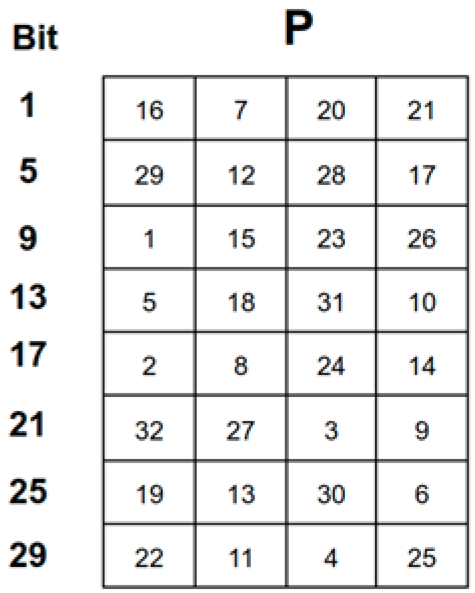
Step-4: P-Box Permutation

The permutation P is defined in the following table.

P yields a 32-bit output from a 32-bit input by permuting the bits of the input block.

This functions similarly to our previous permutations.

Essentially just **shuffling the vector**.



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

Step-5: XOR & Swap

After the permutation in the last phase, we **XOR the 32-Bit LPT with the newly transformed RPT**.

The resulting XOR'd 32-Bit value will be defined as R_1. Let R_0 be the original 32-bit RBT in the equation below.

$$R_1 = L_0 \wedge f(R_0 , K_1)$$

In the next round, we will **swap** L_2 with R_1($L_2 = R_1$).

Then we must calculate

$$R_2 = L_1 \wedge f(R_1, K_2)$$

and so on for 16 rounds.

More About Rounds

Recall the DES overview, there are two ‘sides’ of the process (Left & Right).

These sides are only conceptually separated - each 32-Bit ‘side’ of the Plaintext block are not separately encrypted.

Consider each round as function *f* which operates on two blocks--a data block of 32 bits and a round key K_n of 48 bits - to produce a block of 32 bits.

$$L_n = R_n\text{-}1$$

$$R_n = L_{n-1} \wedge f(R_{n-1}, K_n)$$

Final Permutation

After all 16 Rounds have completed (*n = 16)* we are left with the 64-bit block

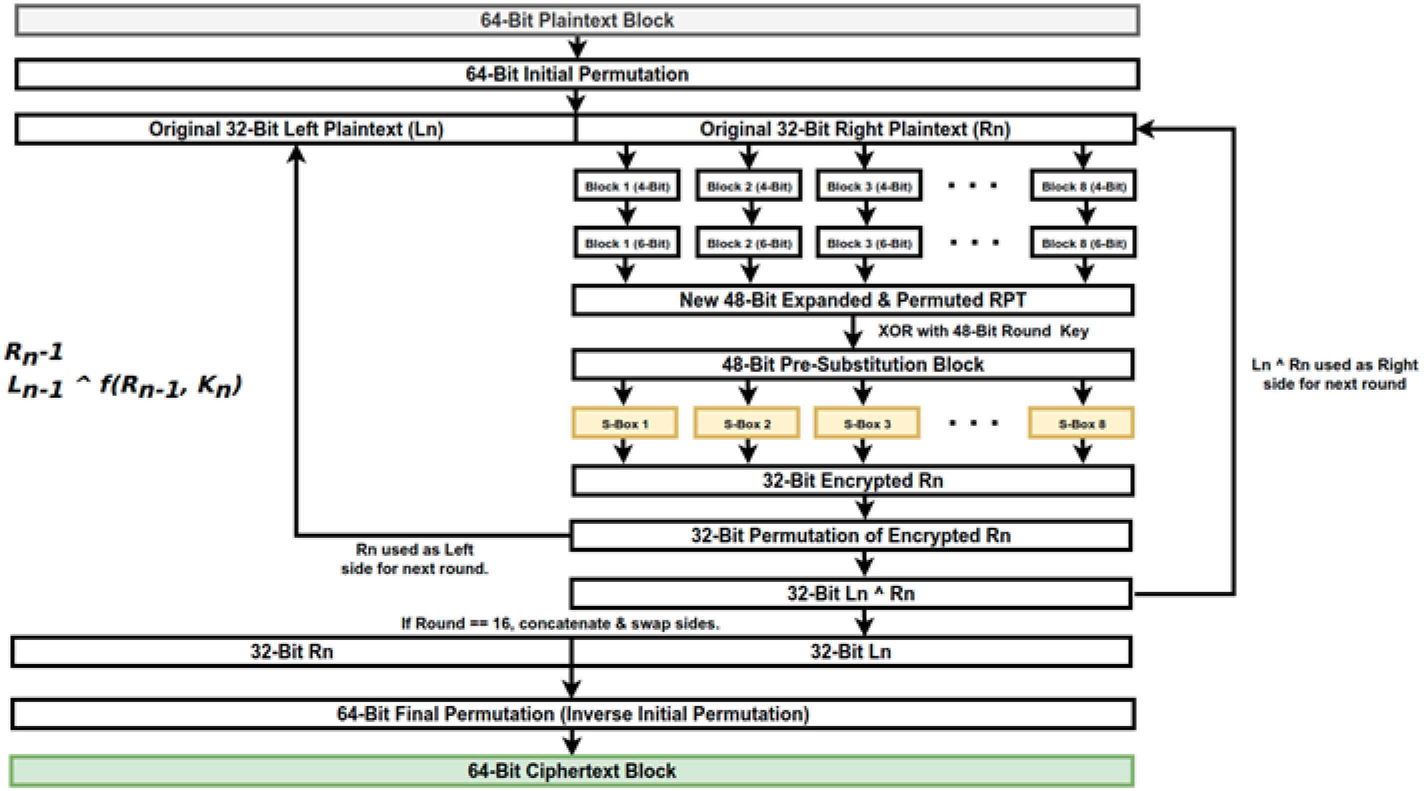
$$L_{16} \mid R_{16}$$

We then reverse the order of the two blocks into the 64-bit block

$$R_{16} \mid L_{16}$$

Finally we take the inverse of the Initial Permutation. (Just undoing the swaps we made at the very beginning).

Put It All Together...


64-Bit Plaintext Block
64-Bit Initial Permutation
Original 32-Bit Left Plaintext (Ln)
Original 32-Bit Right Plaintext (Rn)
Block 1 (4-Bit)
Block 2 (4-Bit)
Block 3 (4-Bit)
Block 8 (4-Bit)
Block 1 (6-Bit)
Block 2 (6-Bit)
Block 3 (6-Bit)
Block 8 (6-Bit)
New 48-Bit Expanded & Permuted RPT
XOR with 48-Bit Round Key
48-Bit Pre-Substitution Block
S-Box 1
S-Box 2
S-Box 3
S-Box 8
32-Bit Encrypted Rn
32-Bit Permutation of Encrypted Rn
Rn used as Left side for next round.
32-Bit Ln ^ Rn
Ln ^ Rn used as Right side for next round
If Round == 16, concatenate & swap sides.
32-Bit Rn
32-Bit Ln
64-Bit Final Permutation (Inverse Initial Permutation)
64-Bit Ciphertext Block

$L_n = R_n\text{-}1$
$R_n = L_{n\text{-}1} \wedge f(R_{n\text{-}1}, K_n)$

Data Encryption Standard

Decryption is simply the inverse of encryption - following the same steps outlined, but reversing the order in which the subkeys are applied.

If each 64-bit Plaintext block is encrypted individually, then the mode of encryption is called Electronic Code Book (ECB) mode.

There are two other modes of DES encryption, namely Chain Block Coding (CBC) and Cipher Feedback (CFB).

These more advanced modes make each cipher block dependent on all the previous blocks by incorporating resulting ciphertext into the encryption process.