2021

Cryptography: DES Implementation

EEGR – 582 – ADVANCED CRYPTOGRAPY (PROJECT 2 – NOTES)

NNAMDI OSUAGWU

MORGAN STATE UNIVERSITY – DR. COLE

Contents

PROJECT CODE – IN GIT

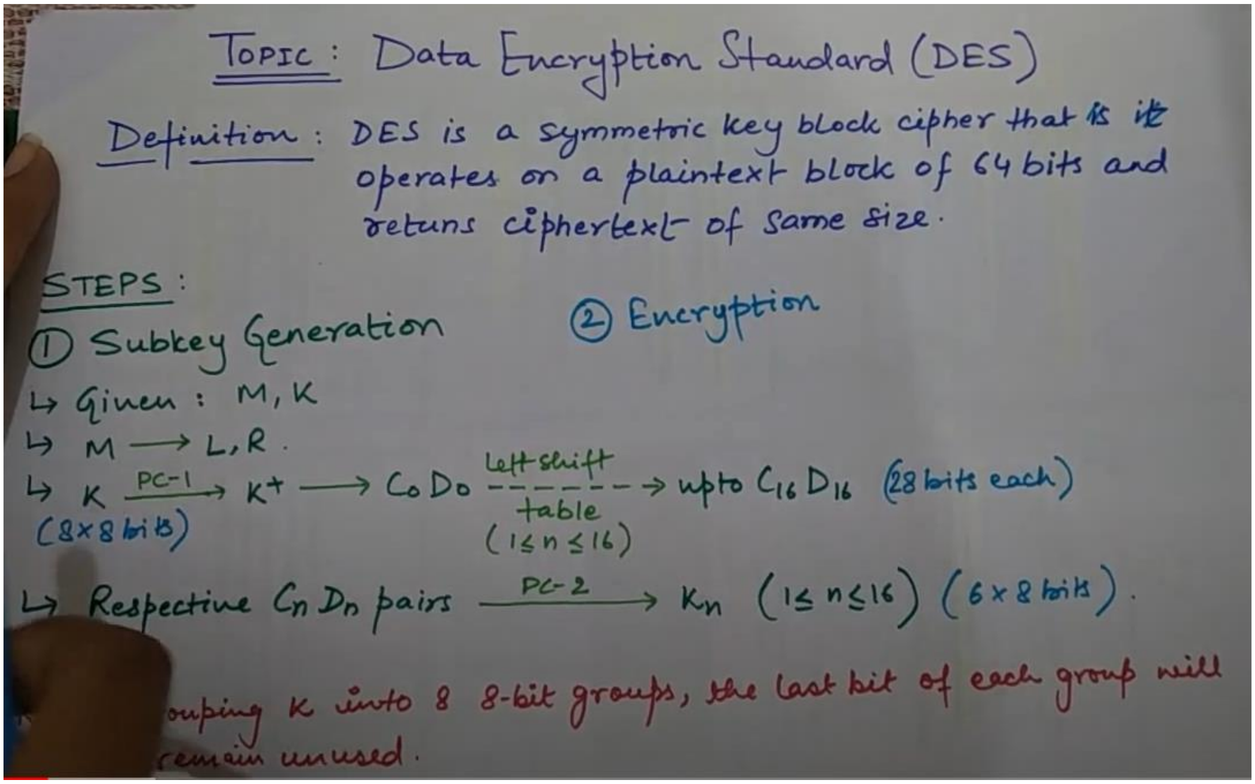
https://github.com/nnamdi2020/Cryptography/blob/main/DESCryptography.py

Project Output

```
PS C:\Users\nosua\OneDrive - Strategic Generation\Certs-School\Morgan State PHD\Spring 2021 Semester\Cryptography
/Cryptography/Project 2/.venv/Scripts/python.exe" "c:/Users/nosua/OneDrive - Strategic Generation/Certs-School/Mo
This program encrypts your plain text using Data Encryption Standard (DES)


Please enter a key (ex: secret_k): (only 8 characters at this time) 12345678


Please enter your plain text (only 8 characters at this time)  (example: iLLmatic): iLLmatic


plain text:  iLLmatic
Ciphered: 'ð\x04\x879Pq!è'
Deciphered:  iLLmatic
```

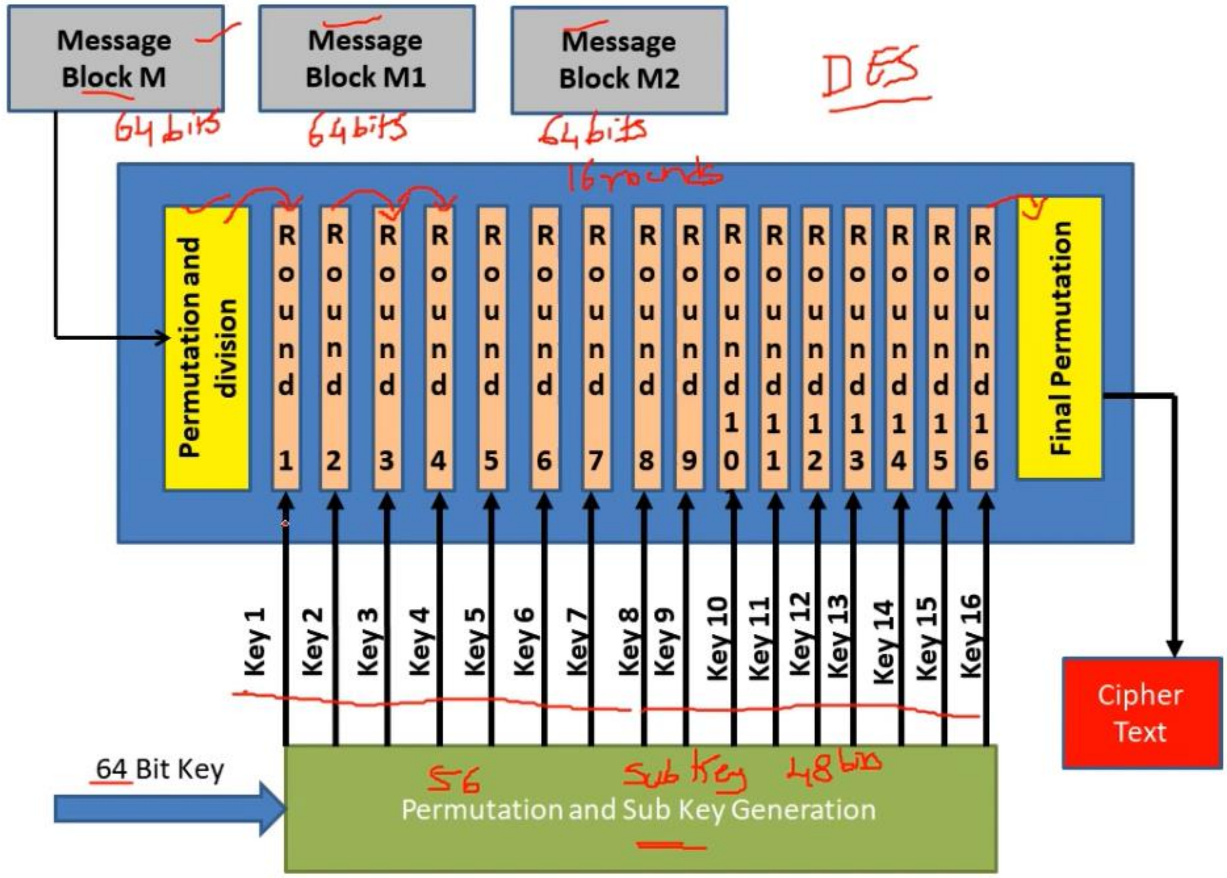
What is DES?

The **DES** (Data **Encryption** Standard) algorithm is a symmetric-key block **cipher** created in the early 1970s by an IBM team and adopted by the National Institute of Standards and Technology (NIST). The algorithm takes the plain text in 64-bit blocks and converts them into ciphertext using 48-bit keys.



I found the following video on DES extremely informative and beneficial to the understanding the project. https://www.youtube.com/watch?v=-j80aA8q_IQ

DES Breakdown



- The message blocks are divided into 64-Bit blocks
- The key is divided into a 56-bit permutation and 16 sub keys (48-bit each).
- 16 rounds perform the same actions. The output of each round is given as input to the next round.

Phase-1 Generating 16 Sub Keys

Key in Hexadecimal = 133457799BBCDFF1 16

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

K is 64 bits (8 x 8 = 64)

Example of conversion: 13: 1 = 0001 , 3 = 0011 , 13 = 00010011

How to convert HEX to Binary

Convert each hex digit to 4 binary digits according to this table:

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

HEX goes up to 16. A – F (10 – 15)

- 2^0 = 1
- 2^1 = 2
- 2^2 = 4
- 2^3 = 8

Converting Hexadecimal to signed 16-bit binary

Converter - https://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html

https://www.rapidtables.com/convert/number/how-hex-to-binary.html

Key in Hexadecimal: 133457799BBCDFF1

Generating 16 Sub Keys

Generating 16 Sub Keys

- The 64-bit key is permuted according to the following table, **PC-1**.
- Note only 56 bits of the original key appear in the permuted key.

64

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111 56 bits

-12

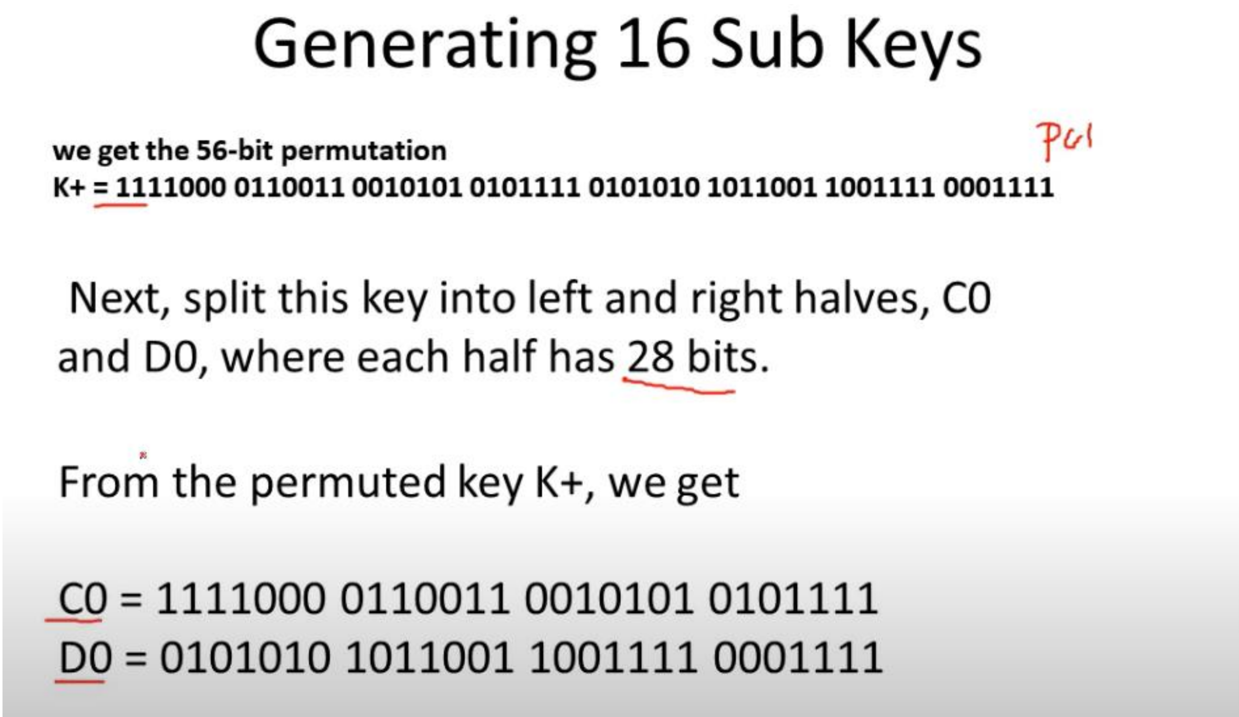
7x8 =56

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

PC-1

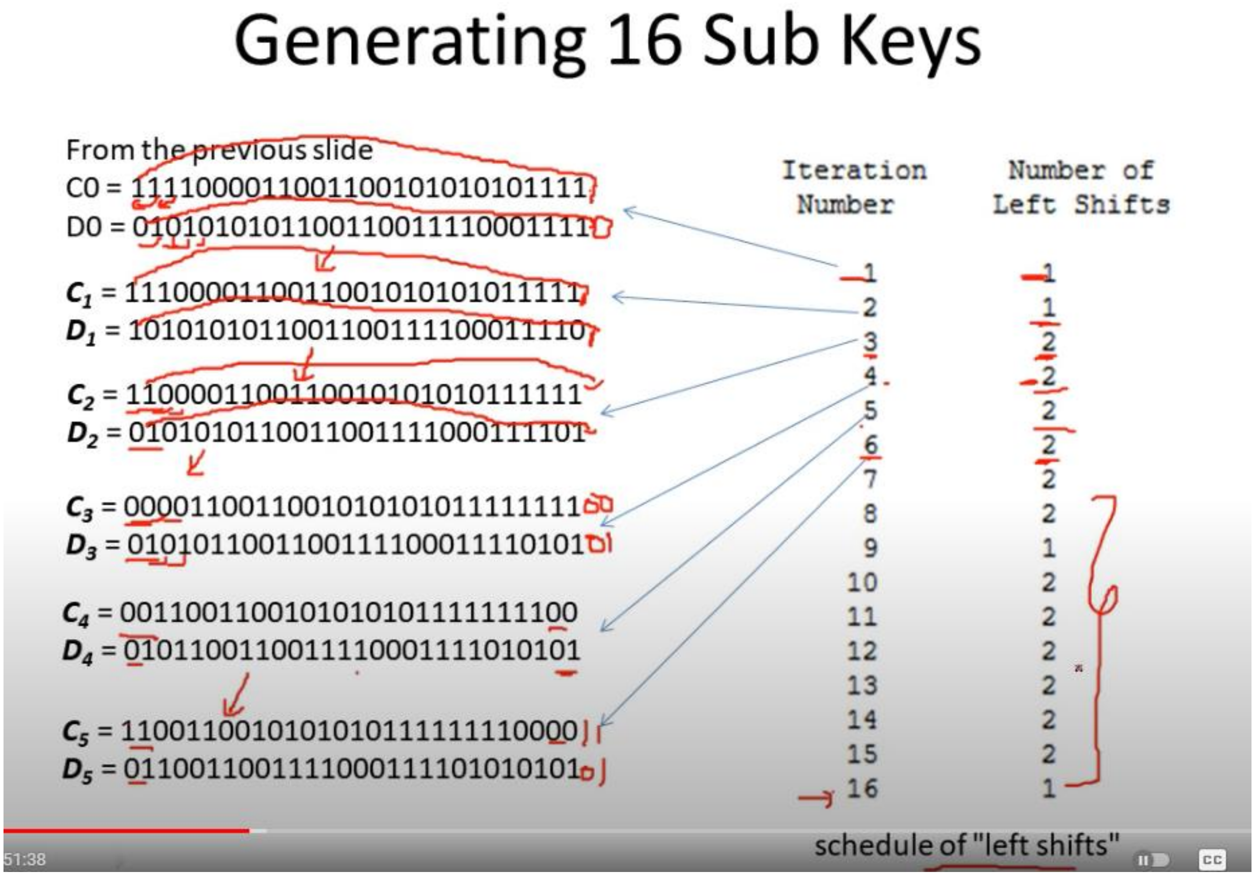
Use all tables (PC-1,PC-2, IP) by going row by row (left to right) . Go to the position of the number (e.g., 57 bit = 1 in K) and place the value in the 1st position of K+.

Split the K+ Key



Left Shifts

Question: How is the Number of Left Shifts generated – is it fixed?



Generating 16 Sub Keys

C_6 = 0011001010101011111111000011
D_6 = 1001100111100011110101010101

C_7 = 1100101010101111111100001100
D_7 = 0110011110001111010101010110

C_8 = 0010101010111111110000110011
D_8 = 1001111000111101010101011001

C_9 = 0101010101111111100001100110
D_9 = 0011110001111010101010110011

C_{10} = 0101010111111110000110011001
D_{10} = 1111000111101010101011001100

C_{11} = 0101011111111000011001100101
D_{11} = 1100011110101010101100110011

C_{12} = 0101111111100001100110010101
D_{12} = 0001111010101010110011001111

C_{13} = 0111111110000110011001010101
D_{13} = 0111101010101011001100111100

C_{14} = 1111111000011001100101010101
D_{14} = 1110101010101100110011110001

C_{15} = 1111100001100110010101010111
D_{15} = 1010101010110011001111000111

C_{16} = 1111000011001100101010101111
D_{16} = 0101010101100110011110001111

Generating the Sub Keys using the PC-2 Table

All of the sub keys are 48 bits

Generating 16 Sub Keys

We now form the keys K_n, for 1<=n<=16, by applying the following permutation table to each of the concatenated pairs C_nD_n.
Each pair has 56 bits, but **PC-2** only uses 48 of these.

C_1 = 1110000110011001010101011111
D_1 = 1010101011001100111100011110

56

C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010 48 bits

1st subkey 48bits

PC-2

8x6 =48

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Cn & Dn are concatenated (56-bits) then the PC-2 table is used to generate a 48-bit Kn

Generating 16 Sub Keys

C_2 = 1100001100110010101010111111
D_2 = 0101010110011001111000111101

C_2D_2 1100001 1001100 1010101 0111111 0101010 1100110 0111100 0111101 56

K_2 = 011110 011010111011011001110110111100100111100101 48bits

subkey2

C_3
D_3

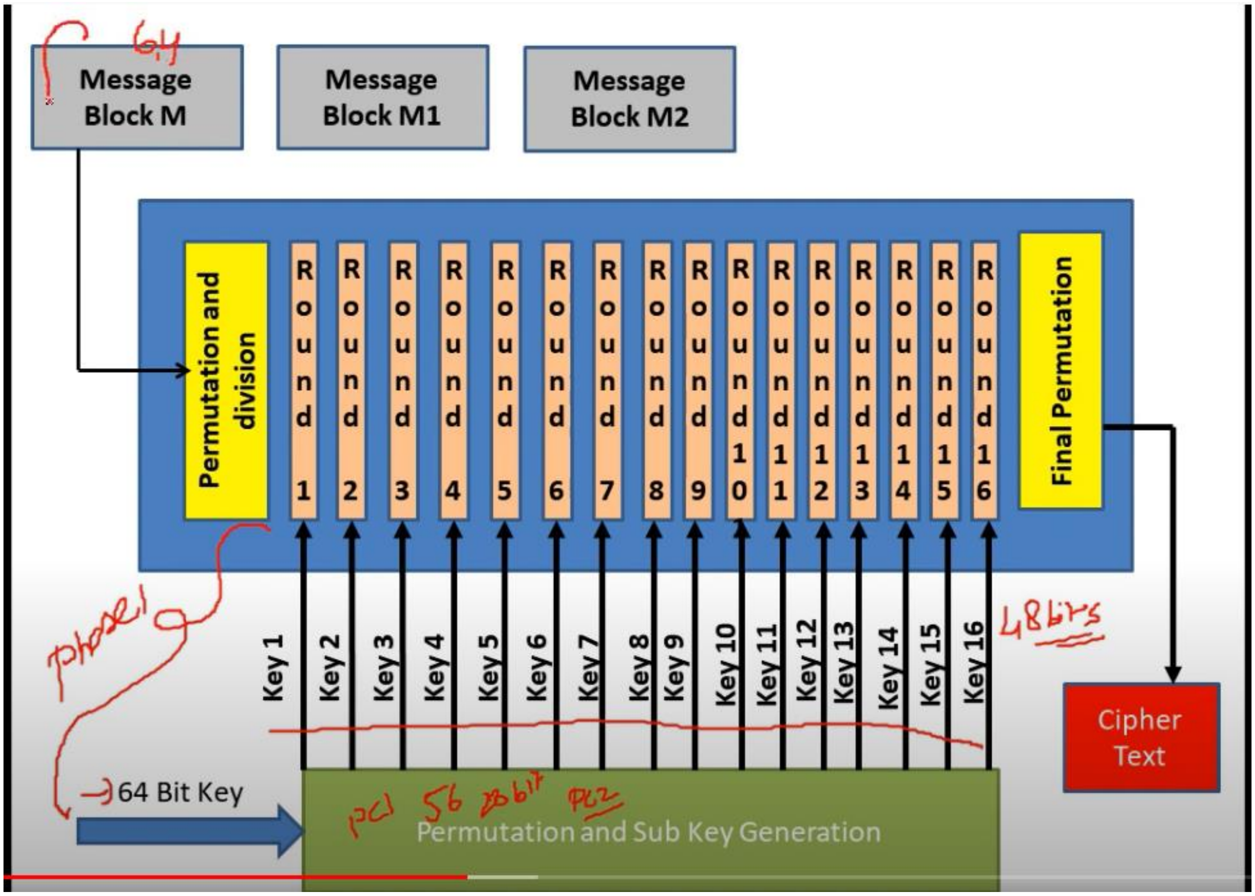
K_3

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Generating 16 Sub Keys

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
K_3 = 010101 011111 110010 001010 010000 101100 111110 011001
K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
K_8 = 111101 111000 101000 111010 110000 010011 101111 111011
K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111
K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101



- We took a 64 bit Key
- applied the PC-1 table to get 56-bit keys (K+)
- Divided K+ it to 28 bits keys C0 D0 keys
- Used left shifts to get C0..16 , D0..16
- Applied the PC-2 table to each C0..16, D0..16 to get K1..16 subkeys . Note K = C & D concatenated with the PC-2 applied to generate a 48-bit key

Phase-2 Permutation and division

Permutation is just rearranging the bits in the message. We will use the IP table

Step 2: Encode each 64-bit block of data.

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

64 bits

There is an *initial permutation* **IP** of the 64 bits of the message data **M**

11

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

IP

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

Divide the IP value (64 Bits) to L0(32 bits) and R0(32 bits)

Step 2: Encode each 64-bit block of data

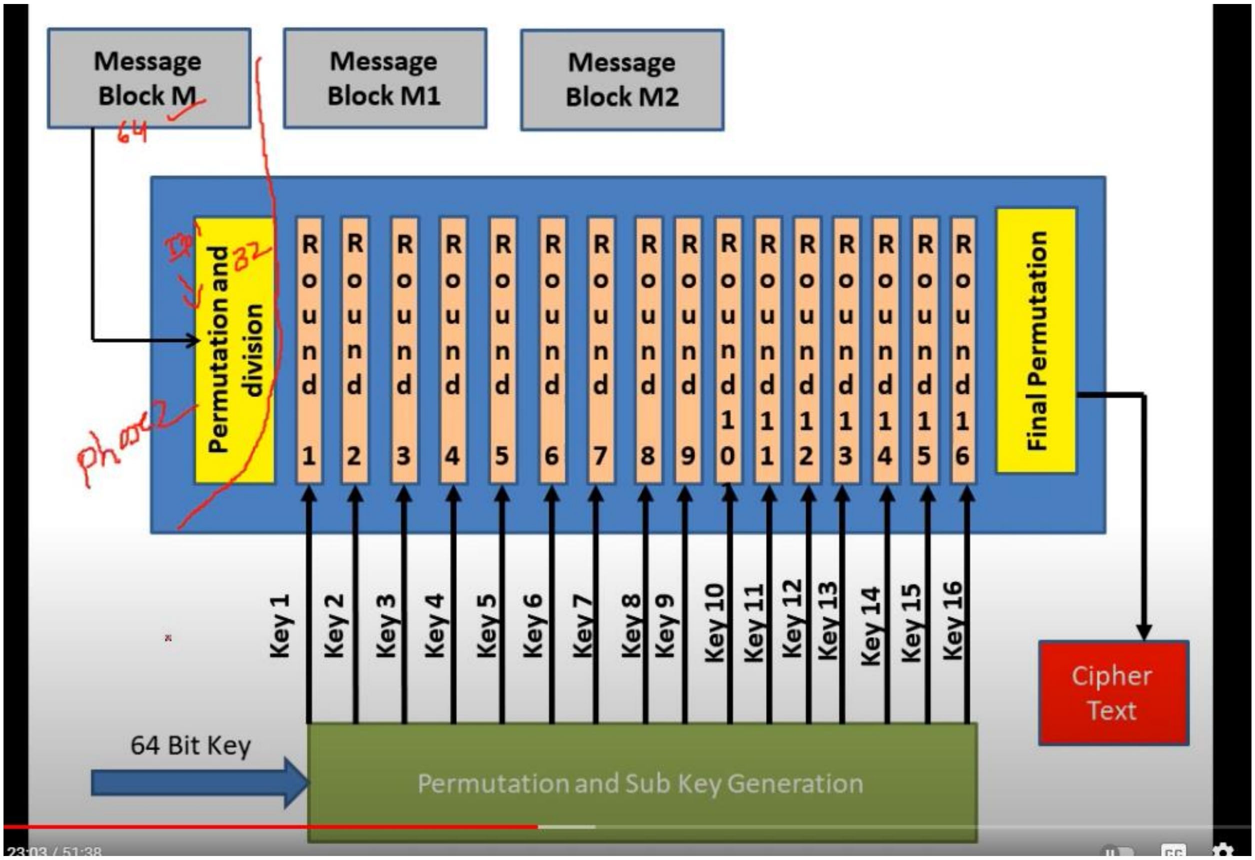
64

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Next divide the permuted block **IP** into a
left half L_0 of 32 bits,
and a right half R_0 of 32 bits.

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111 32 bits
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010 32

Phase 2 completed = applying the IP table to the Message Block and dividing it into 2-32bit values



Phase 3 – Rounds

Same formula for all 16 rounds. Output of Round 1 will be Input of Round 2 and so forth until Round 16

Formula

16 Rounds - Take a look at first round

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111 32 bits
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010 32 bits

$L_n = R_{n-1}$
$R_n = L_{n-1} + f(R_{n-1}, K_n)$ key
Let + denote XOR addition

n=1 for round 1
$L_1 = R_{1-1}$
$R_1 = L_{1-1} + f(R_{1-1}, K_1)$

Output (Round 1)
$L_1 = R_0$
$R_1 = L_0 + f(R_0, K_1)$
Input Round 2
subkey 1

For n = 1, we have
K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$R_1 = L_0 + f(R_0, K_1)$

Need to compute R1(understand f(R0,K1). The following have been previously computed: K1, L1, R0, L0

Step 1 – expand R0 from 32 Bits to 48 bits

What is $\boldsymbol{f}(\boldsymbol{R_0},\boldsymbol{K_1})$?

$\boldsymbol{R_0}$ = 1111 0000 1010 1010 1111 0000 1010 1010

32bits → expand → 48bit

- To calculate $\boldsymbol{f}$, we first expand each block $\boldsymbol{R_0}$ from 32 bits to 48 bits.
- This is done by using a selection table that repeats some of the bits in $\boldsymbol{R_0}$.
- We'll call the use of this selection table the function **E**.
- Thus **E**($\boldsymbol{R_0}$) has a 32 bit input block, and a 48 bit output block.

E($\boldsymbol{R_0}$) = 011110 100001 010101 010101 011110 100001 010101 010101

48 bits

E BIT-SELECTION TABLE

6x8 = 48

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

Perform XOR (+ = XOR*)*

With XOR two of anything = 0. Other than that the answer = 1

Example

1 XOR 1 = 0

0 XOR 0 = 0

1 XOR 0 = 1

0 XOR 1 = 1

$f(R_0, K_1)$

$E(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

Next in the f calculation, we XOR the output $E(R_0)$ with the key K_1:

$K_1 + E(R_0)$.

XOR

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$E(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101
K_1+$E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

The result is 48 bits or we can say eight group containing six bits

S Boxes Computation

$f(R_0, K_1)$

We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**".

K_1+$E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

K1+ E(R0) = ***B1*** ***B2*** ***B3*** ***B4*** ***B5*** ***B6*** ***B7*** ***B8***

We now calculate

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

$f(R_0,K_1)$ - $S_1(B_1)$

K_1+E(R_0) = 011000 010001 011110 111010 100001 100110 010100 100111.

K1+ E(R0) = B1 B2 B3 B4 B5 B6 B7 B8

We now calculate

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

B_1 011000

1100 → 12

00 → 0

0101 → S1(B1)

S1

Column Number

Row No.	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

Where are the S box tables generated? Is it Fixed?

Steps to computing S Boxes

Using B1 = 011000

1. Take the 1st and last bit and XOR [0 0 -> 0] this produces the row 0
2. Take the 4 middle bits [1100 and calculate using base 2]
   a. 2^2 + 2^3 = 4 + 8 = 12 – this produces the column
3. Find the value of the row, column [0,12] in the S1 table = 5
4. Covert 5 to binary (use base 2 , 2^3, 2^2, 2^1,2^0) = 0101
5. S1(B1) = 0101
6. Note there are different S box tables for S1,S2,S3 .. S8

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

S2

$f(R_0,K_1) - S_2(B_2)$

K_1+E(R_0) = 011000 010001 011110 111010 100001 100110 010100 100111.

$K1+ E(R0) =$	***B1***	***B2***	***B3***	***B4***	***B5***	***B6***	***B7***	***B8***

We now calculate

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

B_2 010001 01 − 1 1000 → 8

S2

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

S3

$f(R_0, K_1) - S_3(B_3)$

K_1+E(R_0) = 011000 010001 011110 111010 100001 100110 010100 100111.

K1+ E(R0) = **B1** **B2** **B3** **B4** **B5** **B6** **B7** **B8**

We now calculate

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

B_3	011110
$S_3(B_3)$	1000

00 → 0

1111 → 15

8 — 1000

S3

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

S4

$f(R_0,K_1) - S_4(B_4)$

K_1+E(R_0) = 011000 010001 011110 111010 100001 100110 010100 100111.

K1+ E(R0) = **B1 B2 B3 B4 B5 B6 B7 B8**

We now calculate

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

$S_4(B_4)$ 111010 10 – 2 2 – 0010

$S_4(B_4)$ 0010

1101 → 13

S4

13

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

S5

$f(R_0,K_1)$ – $S_5(B_5)$

K_1+E(R_0) = 011000 010001 011110 111010 100001 100110 010100 100111.

K1+ E(R0) = B1 B2 B3 B4 B5 B6 B7 B8

We now calculate

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

$S_5(B_5)$ 100001

$S_5(B_5)$ 1011

S5

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

S6

$S_6(B_6)$ 100110

$S_6(B_6)$ 0101

S6

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

S7

$S_7(B_7)$ 010100

$S_7(B_7)$ 1001

S7

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

S8

$S_8(B_8)$ 100111

$S_8(B_8)$ 0111

S8

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

$\rightarrow f(R_0,K_1)$ - Final stage of *f*- *Permutation*

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

= 0101 1100 1000 0010 1011 0101 1001 0111

we get

$f(R_0,K_1)$ **= 0010 0011 0100 1010 1010 1001 1011 1011** 32

P

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

Remember why we computed $f(R_0,K_1)$

phase 1 → Round 1

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

$L_n = R_{n-1}$

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

Let + denote XOR addition

n=1 for round 1

$L_1 = R_{1-1}$

$R_1 = L_{1-1} + f(R_{1-1},K_1)$

$L_1 = R_0$

$R_1 = L_0 + f(R_0,K_1)$

For n = 1, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010

$R_1 = L_0 + f(R_0,K_1)$

Finding output of Round 1

$f(R_0,K_1)$ = 0010 0011 0100 1010 1010 1001 1011 1011

L0 = 1100 1100 0000 0000 1100 1100 1111 1111
R0 = 1111 0000 1010 1010 1111 0000 1010 1010

For $\boldsymbol{n}$ = 1, we have

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010 — input for Round 2

$R_1 = L_0 + f(R_0,K_1)$

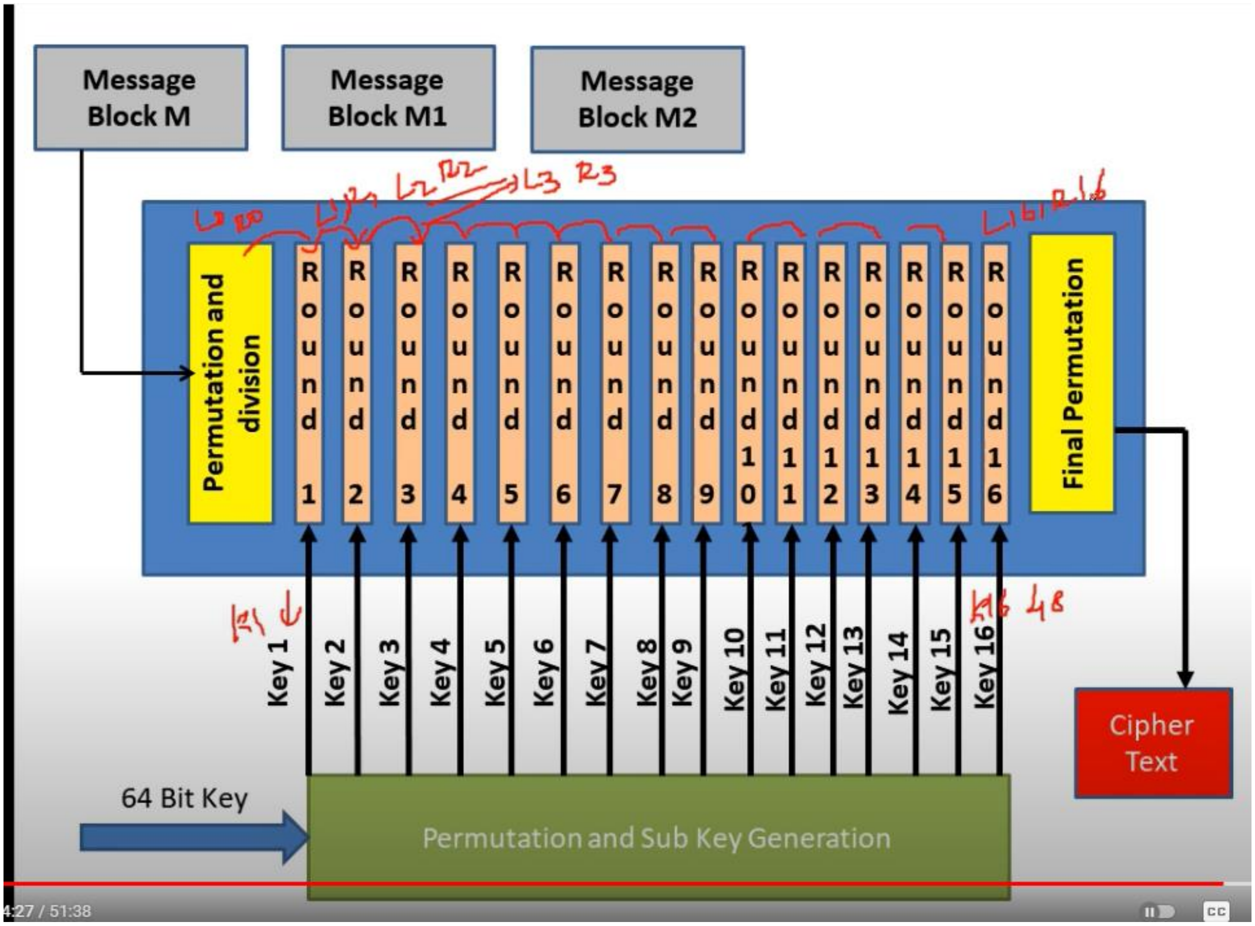
= 1100 1100 0000 0000 1100 1100 1111 1111 L0
+ 0010 0011 0100 1010 1010 1001 1011 1011 f(R0,K1)

R1 = 1110 1111 0100 1010 0110 0101 0100 0100

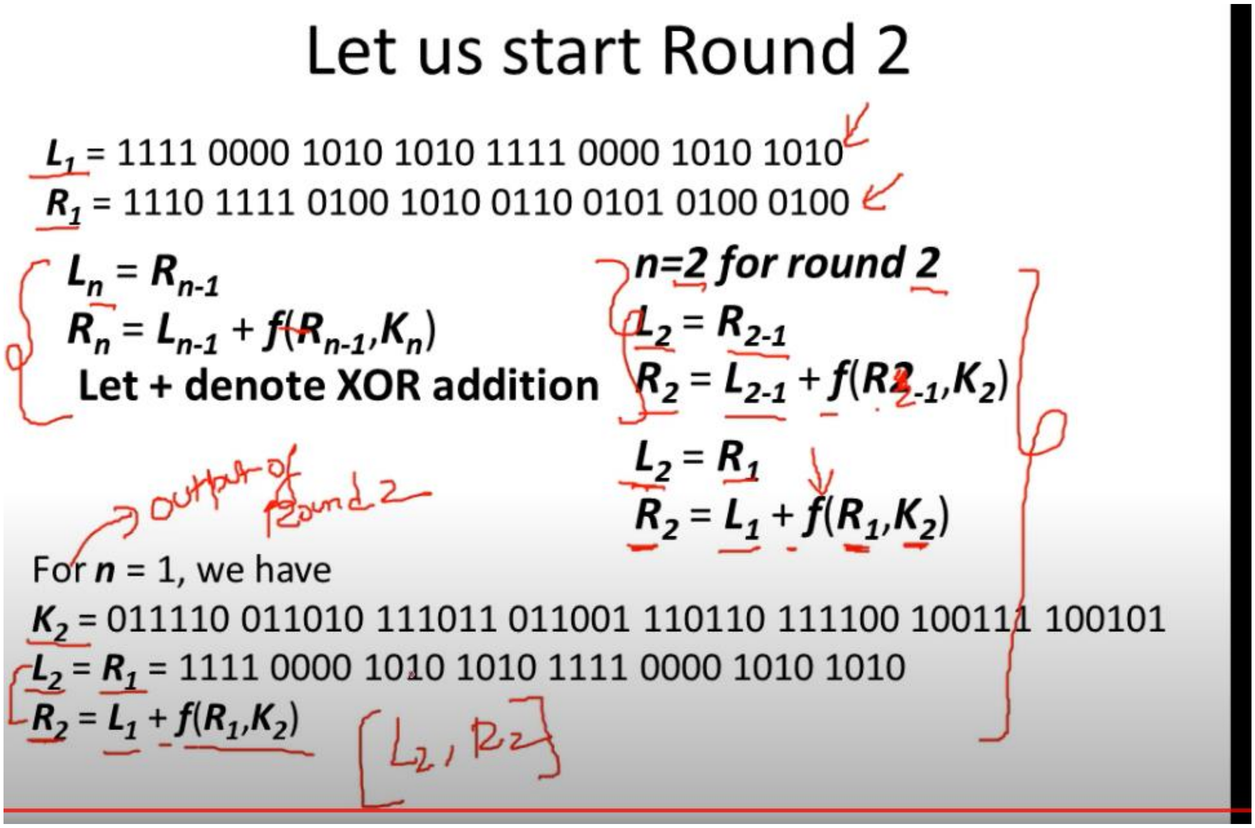
L1 = 1111 0000 1010 1010 1111 0000 1010 1010

32 → output for Round1

Revisit Phase 3

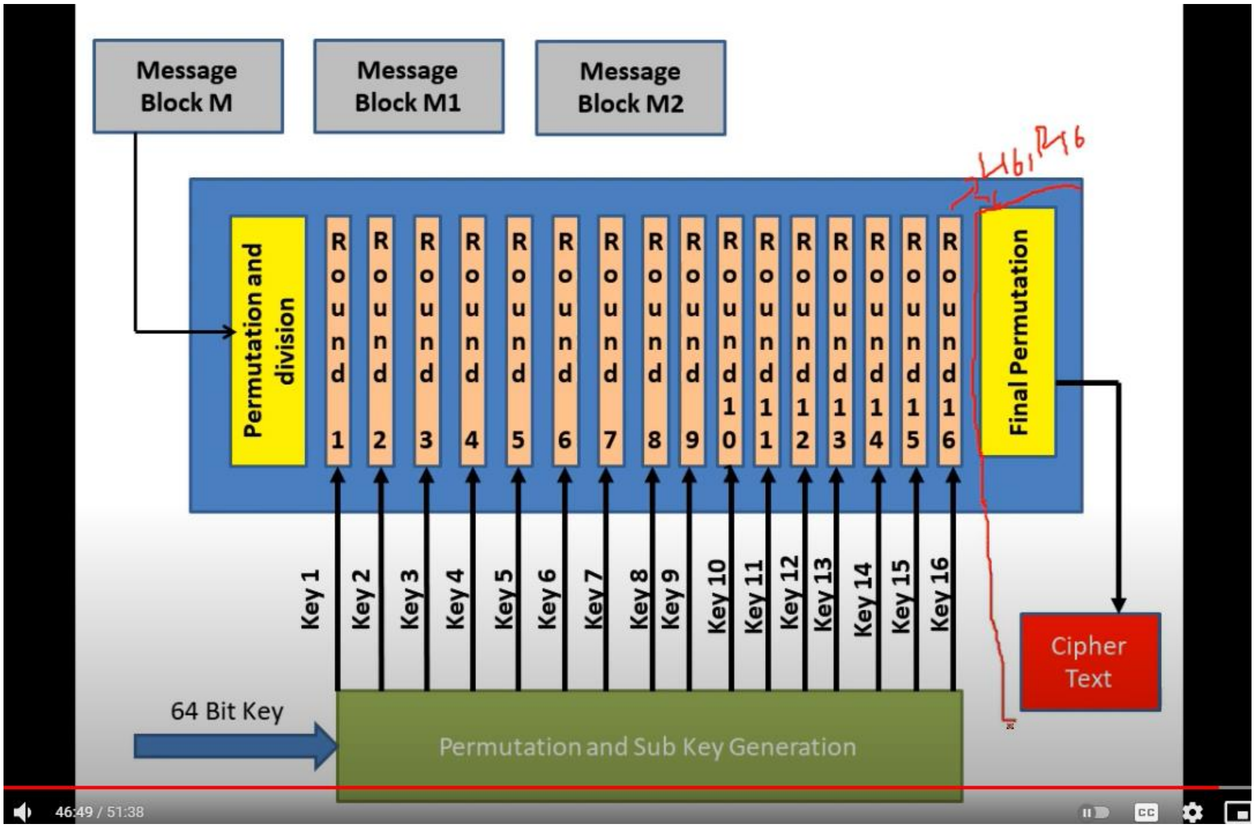


Message Block M
Message Block M1
Message Block M2
Permutation and division
Round 1
Round 2
Round 3
Round 4
Round 5
Round 6
Round 7
Round 8
Round 9
Round 10
Round 11
Round 12
Round 13
Round 14
Round 15
Round 16
Final Permutation
Key 1
Key 2
Key 3
Key 4
Key 5
Key 6
Key 7
Key 8
Key 9
Key 10
Key 11
Key 12
Key 13
Key 14
Key 15
Key 16
Cipher Text
64 Bit Key
Permutation and Sub Key Generation
L2 R2
L3 R3
4:27 / 51:38



Repeat the Above steps for 16 Rounds (Ln, Rn-1, S boxes, etc)

Phase 4 – Final Permutation into Cipher Text



Message Block M
Message Block M1
Message Block M2
Permutation and division
Round 1
Round 2
Round 3
Round 4
Round 5
Round 6
Round 7
Round 8
Round 9
Round 10
Round 11
Round 12
Round 13
Round 14
Round 15
Round 16
Final Permutation
Key 1
Key 2
Key 3
Key 4
Key 5
Key 6
Key 7
Key 8
Key 9
Key 10
Key 11
Key 12
Key 13
Key 14
Key 15
Key 16
Cipher Text
64 Bit Key
Permutation and Sub Key Generation
46:49 / 51:38

Finally after 16 Rounds

Output of 16 Rounds L_{16} R_{16} R_{16} L_{16}

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100 32
R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101 32

We reverse the order of these two blocks and apply the final permutation to
$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101 64 bits

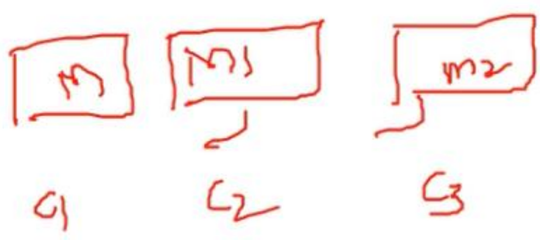
Cyphertext which in hexadecimal format is

85E813540F0AB405.

Cipher text

IP^{-1}

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



Like that we encrypt every block of message

The Truth

64 → 56

- DES is insecure due to the relatively short 56-bit key size.
- In January 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes
- This cipher has been superseded by the Advanced Encryption Standard (AES).
- DES has been withdrawn as a standard by the National Institute of Standards and Technology.