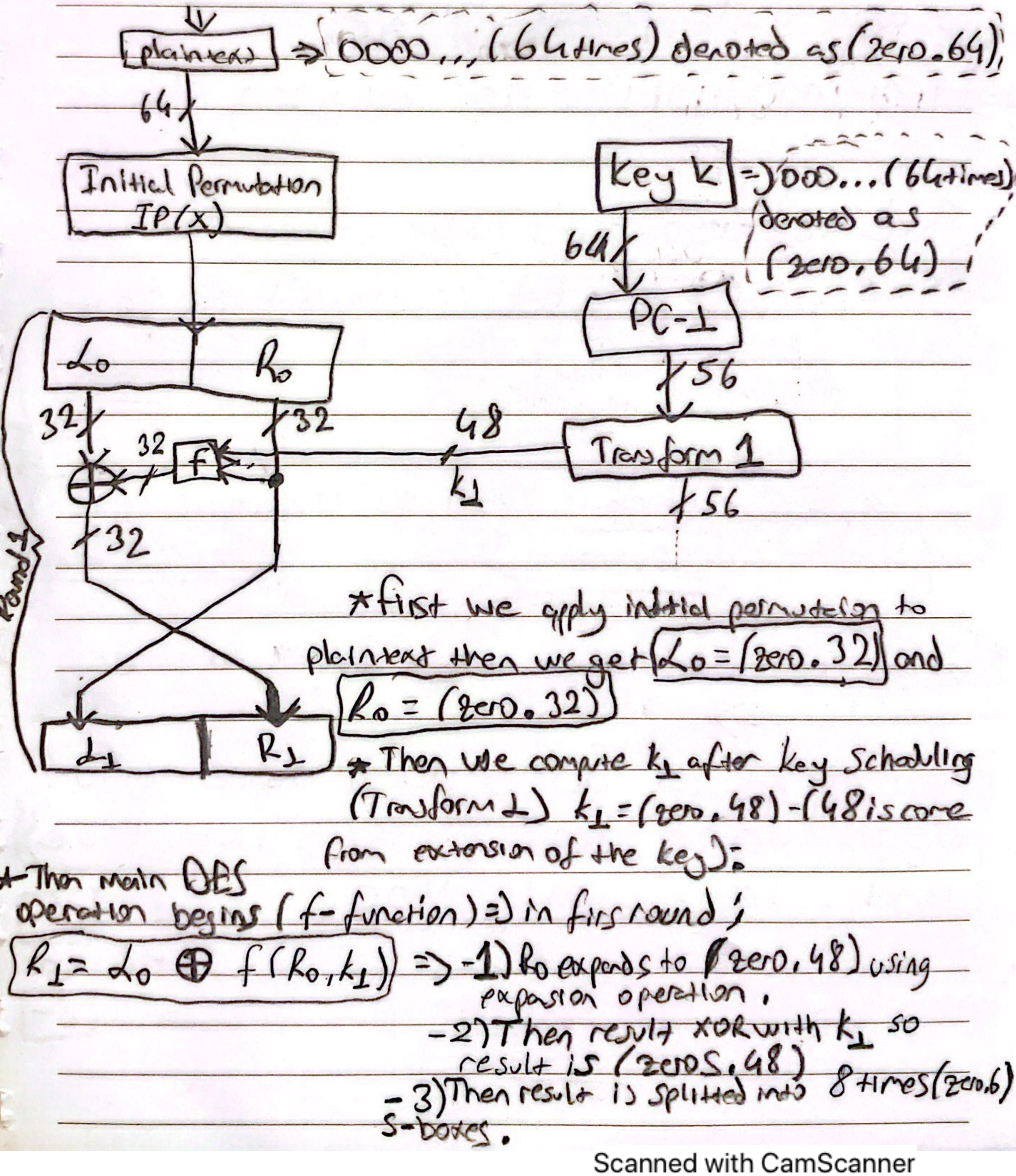
TED ÜNİVERSİTESİ

__/__/__

① What is the Output of the first round of the "DES" algorithm when the plaintext and the Key are both all zeros?



*first we apply initial permutation to plaintext then we get $L_0 = (zero.32)$ and $R_0 = (zero.32)$

*Then we compute k_1 after key Scheduling (Transform 1) $k_1 = (zero.48)$ - (48 is come from extension of the key);

*Then main DES operation begins (f-function) ⇒ in first round;

$k_1 = L_0 \oplus f(R_0, k_1)$ ⇒ -1) R_0 expands to (zero.48) using expansion operation.

-2) Then result XOR with k_1 so result is (zero.48)

-3) Then result is splitted into 8 times (zero.6) S-boxes.

Then we obtain "8" 4-bit values after applying S-box substitution:

⇒ 1110 1111 1010 0111 0010 1100 0100 1101

Lastly, we need to apply permutation operation according to the p-table values:

⇒ 1101 1000 1101 1000 1101 1011 1100 ⇒ After p-table conversion

⇒ Since we crossover the results at the end of the DES. $R_1 = L_0 \oplus f(R_0, k_1)$ is corresponds the right 32-bit.

↓ all zero

↑ we found it above recently.

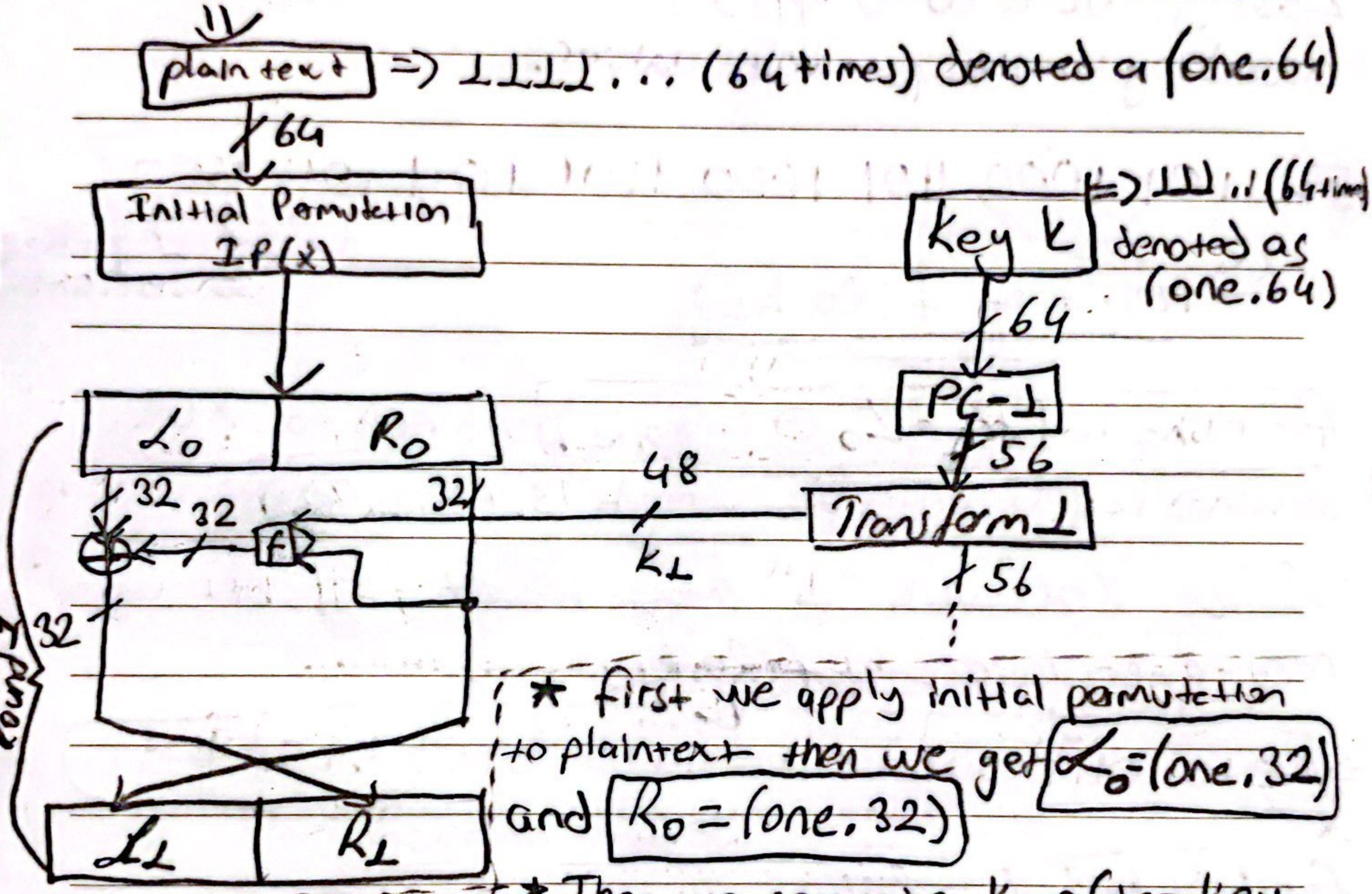
$L_1 = R_0$ → which is (zero, 32) is correspond the left 32-bit.

*So 64-bit final result by putting together them is ⇒

0000 0000 0000 0000 0000 0000 0000 0000
1101 1000 1101 1000 1101 1011 1100

Answer

② What is the output of the first round of the DES Algorithm when the plaintext and the key are both all ones?



* first we apply initial permutation to plaintext then we get L_0=(one.32) and R_0=(one.32)

* Then we compute K_1 after key scheduling (Transform 1) K_1=(one.48)

48 is come from extension of the key,

* Then main DES operation begins (f-function) in first round;

$R_1 = L_0 \oplus f(R_0, K_1)$ => [1] R_0 expands to (one.48) using expansion operation.

[-2] Then result XOR with K_1 so result is (zero.48).

[-3] Then result is splitted into 8 times (zero.6) S-boxes.

TED ÜNİVERSİTESİ

__/__/____

After S-box substitution we obtain "8" 4-bit values.

=> 1110 1111 1010 0111 0010 1100 0100 1101

Lastly, we need to apply permutation operation according to the p-table values:

=> 1101 1000 1101 1000 1101 1011 1011 1100

↳ After p-table conversion

↳ result of $f(R_0, k_1)$

for right half $R_1 = L_0 \oplus f(R_0, k_1)$ we need to XOR obtained result with "L_0" which is (one, 32)

=> Since XOR with "1" means inverting all bit we need to get inverse of $f(R_0, k_1)$

=> 0010 0111 0010 0111 0010 0100 0100 0011

↳ After inversion

for left half $L_1 = R_0$ (which is (one, 32))

So, 64 bit final result by putting together them is =>

1111 1111 1111 1111 1111 1111 1111 1111

0010 0111 0010 0111 0010 0100 0100 0011

Answer

__/__/__

(3) What is the output of the first round of DES algorithm when the plaintext is first 64 bits of your name and surname and the key is the first 64 bits of your student number?

Name, Surname: 01000010 01000001 01001000 01000001 01000100 01001001 01010010 11011100 → our plaintext

Student Number: 00110001 00110000 00110000 00110000 00110111 00110111 00110110 00111000 → our key

* first we need to apply (Initial permutation) to plaintext:

11111111 11000000 10011000 → L_0 part

00101010 10000000 00000000

01001100 01000001 → R_0 part

* Then we compute K_1 after key scheduling (Transform 1):

Key Scheduling steps: * Initially parity bits are removed by PC-1.

$K^+ \Rightarrow$ 0101011 1100000 1101001 0010000 0000100 1010000 0100001 0010101

Next, we will split the permuted key (56 bits) into left and right halves as C_0 and D_0

C_0 = 0101011 1100000 1101001 0010000

D_0 = 0000100 1010000 0100001 0010101

→ Then we need to transform it to "48 bit" to use this as a key "K_1".

TED ÜNİVERSİTESİ

__/__/____

After apply permutation PC-2

⇒ K_1 = 000100 001110 000101 110001 010000 000010 110100 000100

* Then main DES operation begins (f-function)

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

1) R_0 needs to expand to 48 bit inorder to increase diffusion.

⇒ R_0 => 10000000 00000000 10100100 01000001

⇓ Expansion operation to 48-bit.

R_0^{48} = 110000 000000 000000 000001 010100 001000 001000 000011

- Then we XOR 48 bit expaded R_0 with K_1,

⇒

K_1 = 000100 001110 000101 110001 010000 000010 110100 000100

R_0^{48} = 110000 000000 000000 000001 010100 001000 001000 000011

$K_1 \oplus R_0^{48}$ = 110100 001110 000101 110000 000100 001010

B_1 B_2 B_3 B_4 B_5 B_6

111100 000111

B_7 B_8

* Then result is splitted into 8 times 6-bit S-boxes.

$K_1 \oplus R_0^{48}$ = (.....)

$S_1(B_1) \Rightarrow (110100) S_1 \Rightarrow$ 1001, $S_2(B_2) \Rightarrow (001110) S_2 \Rightarrow$ 0100

$S_3(B_3) \Rightarrow (000101) S_3 \Rightarrow$ 0000, $S_4(B_4) \Rightarrow (110000) S_4 \Rightarrow$ 1111

$S_5(B_5) \Rightarrow (000100) S_5 \Rightarrow$ 0100, $S_6(B_6) \Rightarrow (001010) S_6 \Rightarrow$ 0010

$S_7(B_7) \Rightarrow (111100) S_7 \Rightarrow$ 1001, $S_8(B_8) \Rightarrow (000111) S_8 \Rightarrow$ 1000

TED ÜNİVERSİTESİ

0011
S6 A8

/ /

⇒ for the first round, we obtain as the output of the eight S-boxes:

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

1001 0100 0000 1111 0100 0010 1001 1000 → S-boxes output

* The final stage in the calculation of "f" is to do a permutation P of the S-box output to obtain the final value of "f".

So ⇒ "P(S-boxes output)" we get result of "f" as

→ f = 1000 1010 1110 0100 0001 0000 0100 0111 XOR

$L_1 = L_0 \oplus f(R_0, k_1)$ = 1111 1111 1100 0000 1001 0000 00101010 ← P

1000 1010 1110 0100 0001 0000 0101 0111

⇒ 0011 0101 0010 0100 1000 0000 0111 1101

for right half. ← R1 ← ⊕

⊛ for left half we need L_1 which is equal to R_0 directly. Since we know R_0 output of the first round is

⇒ $L_1 + R_1$

⇒ 1000 0000 0000 0000 1010 0100 0100 0001 0011 0101 0010 0100 1000 0000 0111 1101

Answer.

TED ÜNİVERSİTESİ

04 When we flipped the first bit in the 64-bit "plain key":

*Since "PC-1" map 8th key with 1st key of the plain key, Our fliped bit goes to 8th bit of the 56-bit permuted key"

*Then we split 56-bit key into left and right halves as "C" and "D" which are "28 bit each.

Now we derieving 16 key pairs by applying pre-determined shifting operations:

*In rounds "1, 2, 9, 16", two halves are rotated left by "one bit".

*In all other rounds they are roteted left by two bit.

**Since the total number of rotations is "4x1 + 12x2=28" we shif left and right halves "28 time" left.
This means that first and second half before permutation and first and second half after permutation is equal.

*So we can igrore the right half of the permuted bits which is "D". We only need to trace S-boxes in the left part of the permuted bit.

So we can calculate which S-boxes in which rounds are affected step by step:

for ROUND 1; After "1 position" left shift bit is in the "7th" place → "20th" which is affected by the S-box "4" by applying PC-2 position will be.

⇒ After describing some iterations in detail, we'll write affected ones as a table to make explanation clearer.

In "ROUND 2", Position "6" (After "1" left shift) $\xrightarrow{PC-2}$ Position "10" (After PC-2)
↳ Affected S-box: "2"

In "ROUND 3" Position "4" (After "2" left shift) $\xrightarrow{PC-2}$ Position "16" (After PC-2)
↳ Affected S-box: "3"

* I'm continued with tabular form to show process clearly *

Round	Position After left shift	PC-2 →	Position after PC-2	S-box
4	2	"	24	4
5	28	"	8	2
6	26	"	17	3
7	24	"	4	1
8 → eliminated at PC-2, since it has no effect.				
9	21	"	11	2
10	19	"	14	3
11	17	"	2	1
12	15	"	9	2

TED ÜNİVERSİTESİ

Round	Position After Left Shift	PC-2 →	Position After PC-2	S-Box
13	13		23	4
14	11		3	1
15	→ eliminated at PC-2, since it has no effect.			
16	8		18	3

Answer