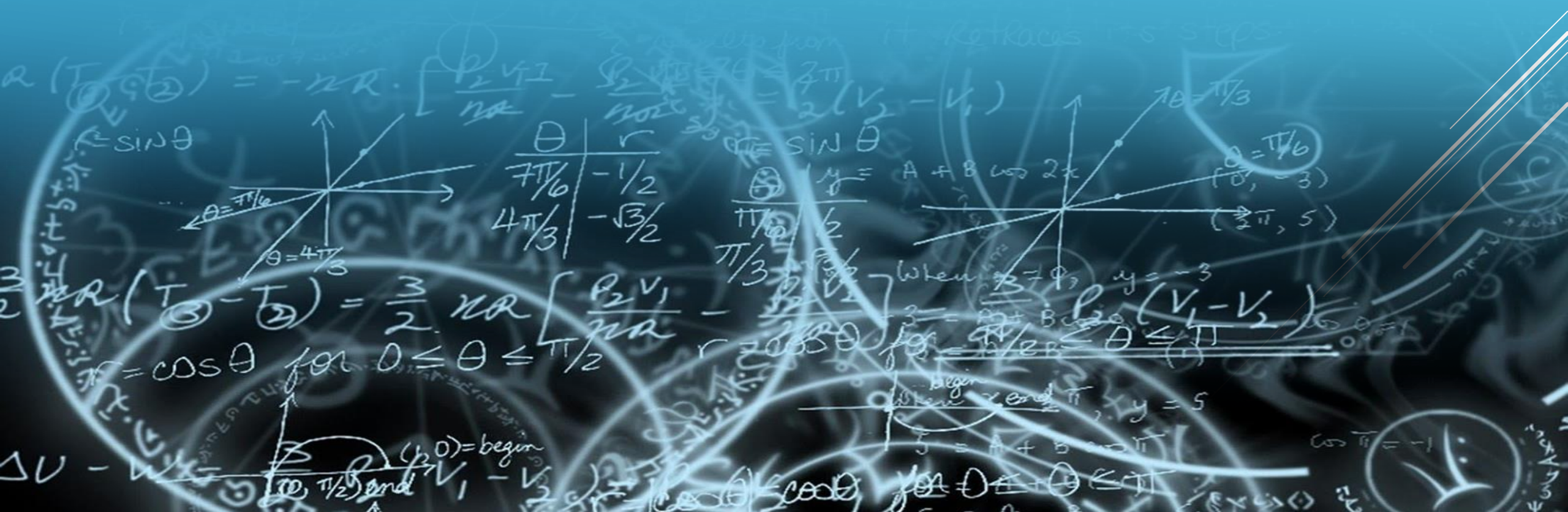


ALGORITMUL DES



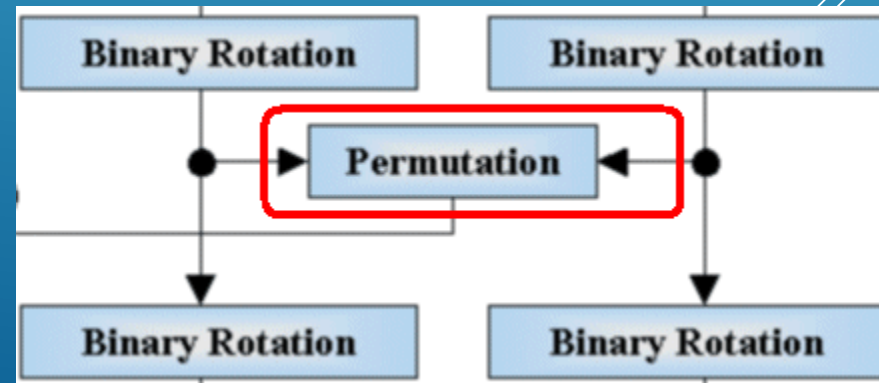
Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

P1.4

Vom calcula sub-cheile K_n $n=1,16$ aplicând următoarea permutare celor 16 numere binare obținute prin concatenarea C_nD_n :

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



Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Ex:

$C_1D_1 = 1110000\ 1100110\ 0101010\ 1011111\ 1010101\ 0110011\ 0011110\ 0011110$

$K_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$

Celelalte chei vor fi în ordine:

$K_2 = 011110\ 011010\ 111011\ 011001\ 110110\ 111100\ 100111\ 100101$ (6 * 8 = 48b)

$K_3 = 010101\ 011111\ 110010\ 001010\ 010000\ 101100\ 111110\ 011001$

$K_4 = 011100\ 101010\ 110111\ 010110\ 110110\ 110011\ 010100\ 011101$

$K_5 = 011111\ 001110\ 110000\ 000111\ 111010\ 110101\ 001110\ 101000$

$K_6 = 011000\ 111010\ 010100\ 111110\ 010100\ 000111\ 101100\ 101111$

$K_7 = 111011\ 001000\ 010010\ 110111\ 111101\ 100001\ 100010\ 111100$

$K_8 = 111101\ 111000\ 101000\ 111010\ 110000\ 010011\ 101111\ 111011$

$K_9 = 111000\ 001101\ 101111\ 101011\ 111011\ 011110\ 011110\ 000001$

$K_{10} = 101100\ 011111\ 001101\ 000111\ 101110\ 100100\ 011001\ 001111$

$K_{11} = 001000\ 010101\ 111111\ 010011\ 110111\ 101101\ 001110\ 000110$

$K_{12} = 011101\ 010111\ 000111\ 110101\ 100101\ 000110\ 011111\ 101001$

$K_{13} = 100101\ 111100\ 010111\ 010001\ 111110\ 101011\ 101001\ 000001$

$K_{14} = 010111\ 110100\ 001110\ 110111\ 111100\ 101110\ 011100\ 111010$

$K_{15} = 101111\ 111001\ 000110\ 001101\ 001111\ 010011\ 111100\ 001010$

$K_{16} = 110010\ 110011\ 110110\ 001011\ 000011\ 100001\ 011111\ 110101$

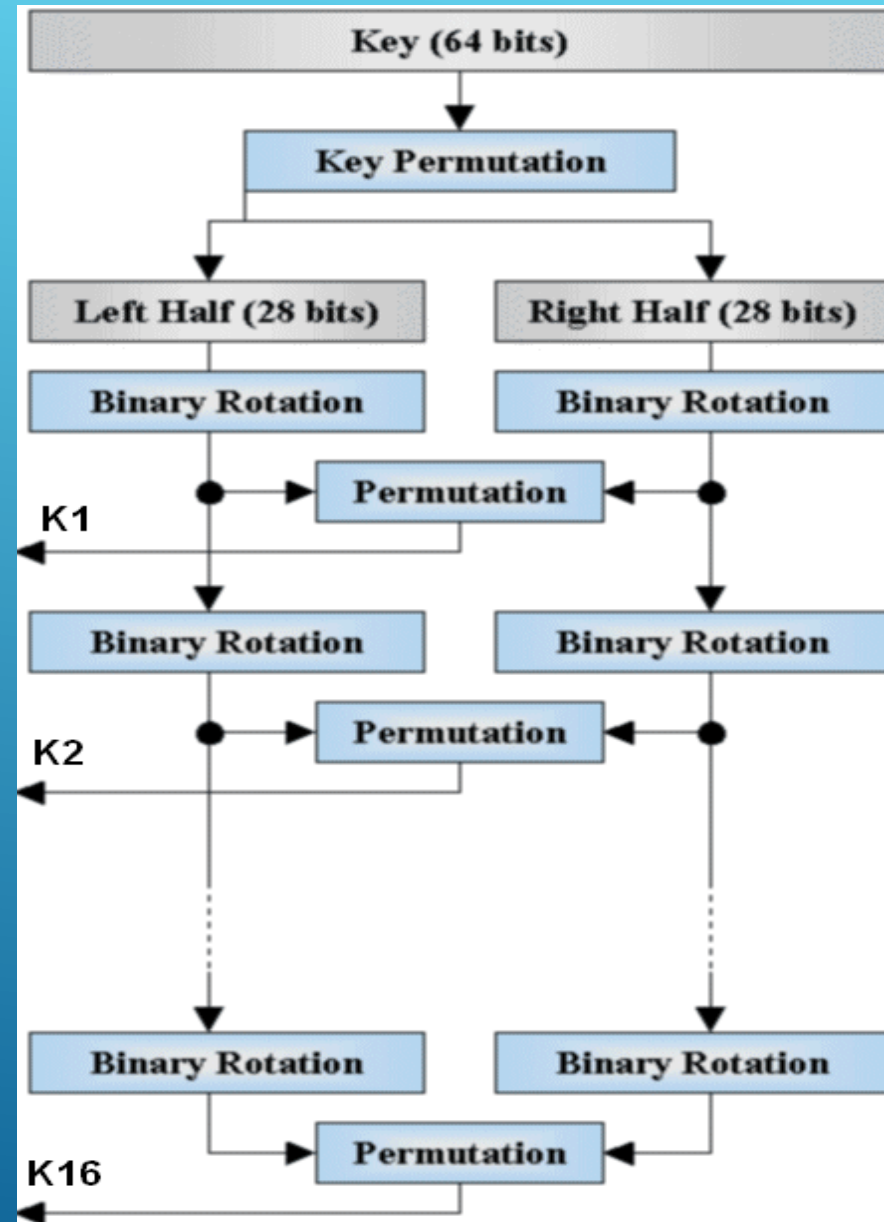
Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Prelucrarea cheilor, funcții de implementat:

1. DES_Key_Permut_PC1();
2. DES_Key_Divide_C0_D0();
3. DES_Key_Divide_Ci_Di();
4. DES_Key_Permut_PC2();
5. DES_Key_CreateSubKeys()

care va apela toate funcțiile anterioare.



Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

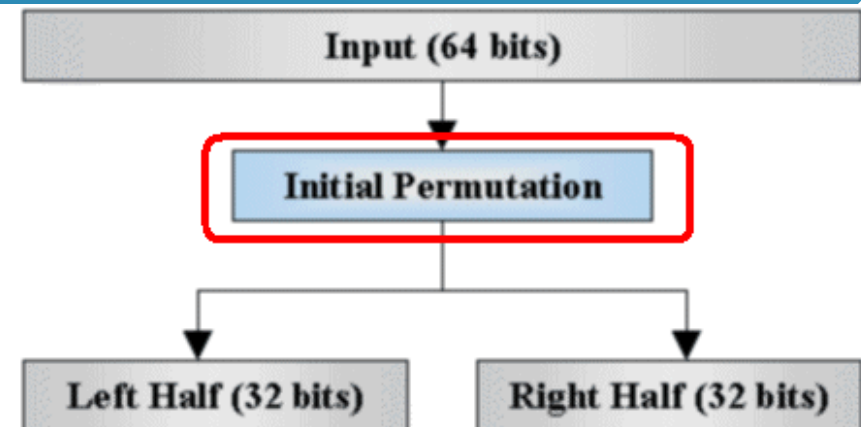
Pasul 2 – Criptarea fiecărui grup de 64 de biți

P2.1

Vom efectua o permutare asupra grupului nițial de 64 de biți pe care o vom numi **IP (initial permutation)**:

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

Initialization



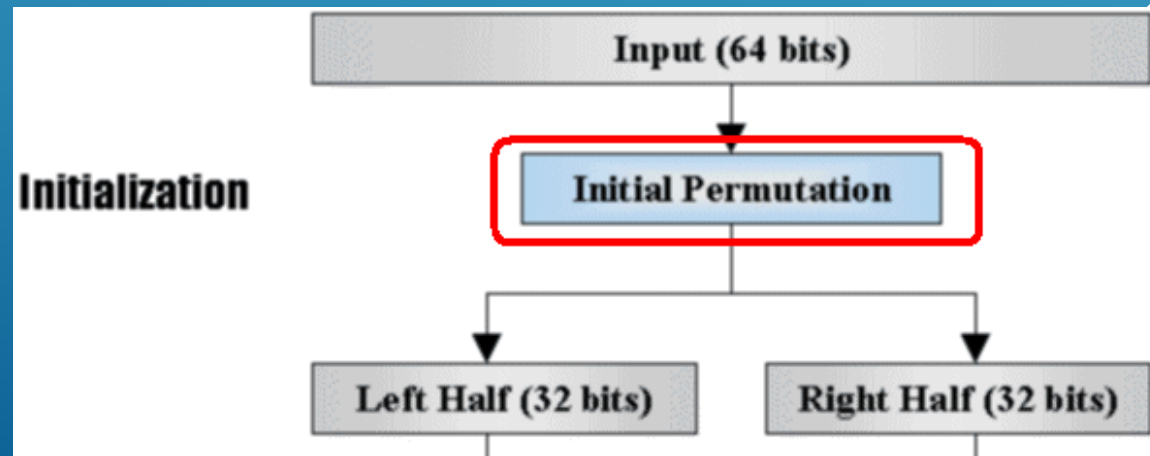
Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Ex:

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

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



Algoritmul de criptare DES (Data Encryption Standard)

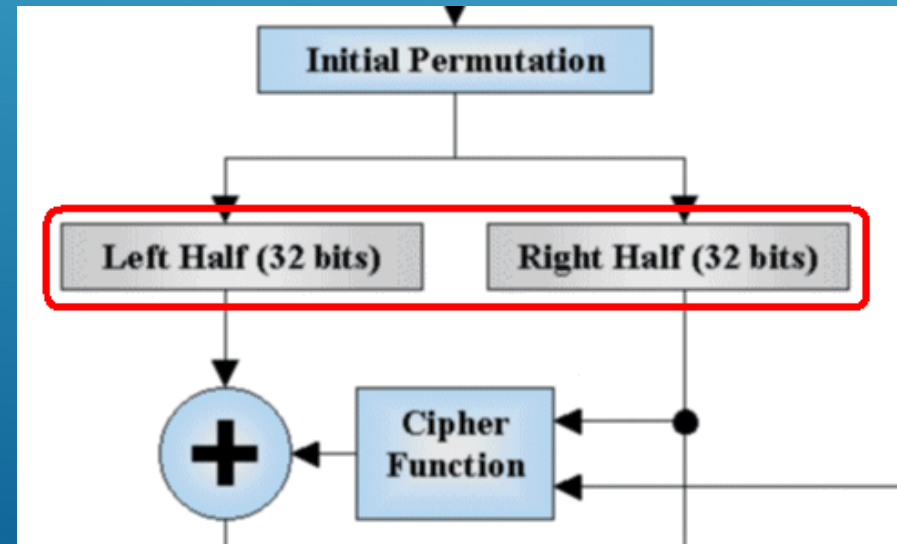
3. DES, descriere în detaliu

P2.2

Vom împărți permutația IP în două jumătăți de câte 32 de biți fiecare: L0 și R0 (left, right):

$L0 = 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111$

$R0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$



Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

P2.3

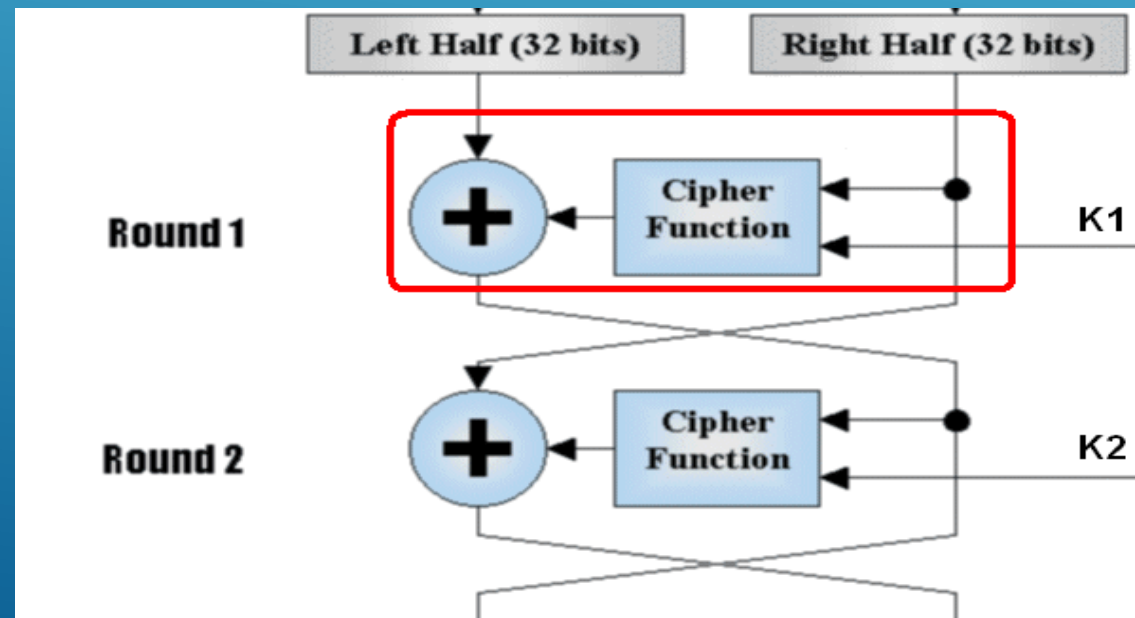
Trecând prin 16 iterații și utilizând o funcție f ce acționează asupra a două blocuri, unul de date de 32 de biți și unul de subcheie de 48 de biți, și va produce un rezultat de 32 de biți.

Vom nota cu $+$ operația pe biți XOR.

În iterații vom efectua următoarele calcule (fără runda 1):

$$L_n = R_{n-1}$$

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

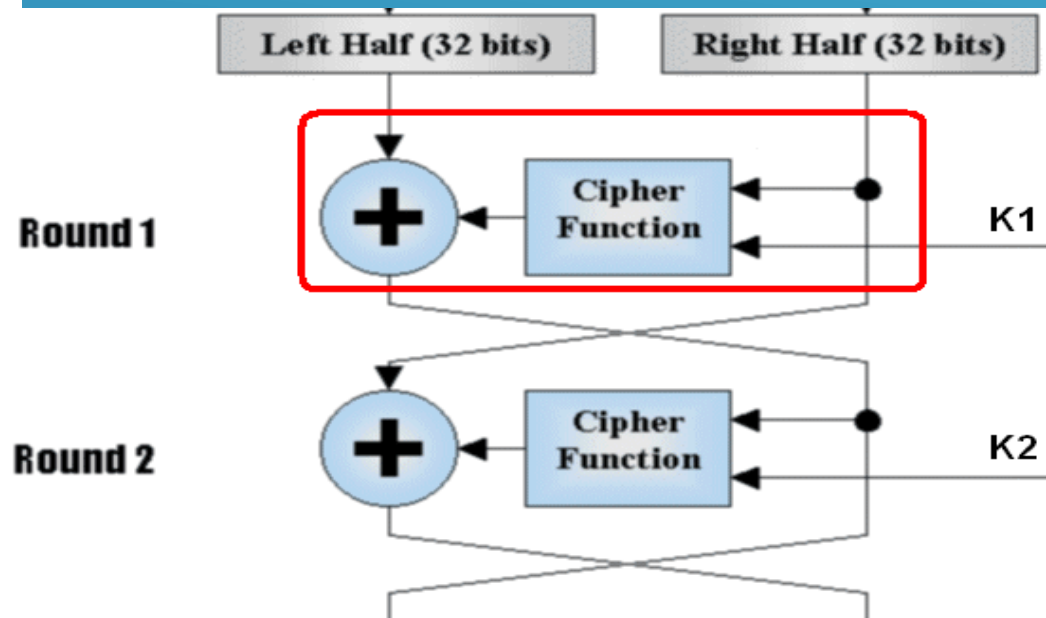


Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Pentru a calcula $f(R_{n-1}, K_n)$ va trebui să expandăm fiecare bloc de 32 de biți la unul de 48 folosind tabela de permutări E:

E	BIT-SELECTION TABLE				
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



Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Obs: Rezultatul ultimei operații XOR este un număr pe 48 de biți. $R1$ din formula $R1 = L0 + f(R0, K1)$ este însă pe 32 de biți. Pentru a obține acest număr de 32 de biți vom trece din nou printr-o serie de permutații.

Vom scrie $Kn + E(Rn-1) = B1B2B3B4B5B6B7B8$, unde fiecare Bi este un grup de 6 biți.

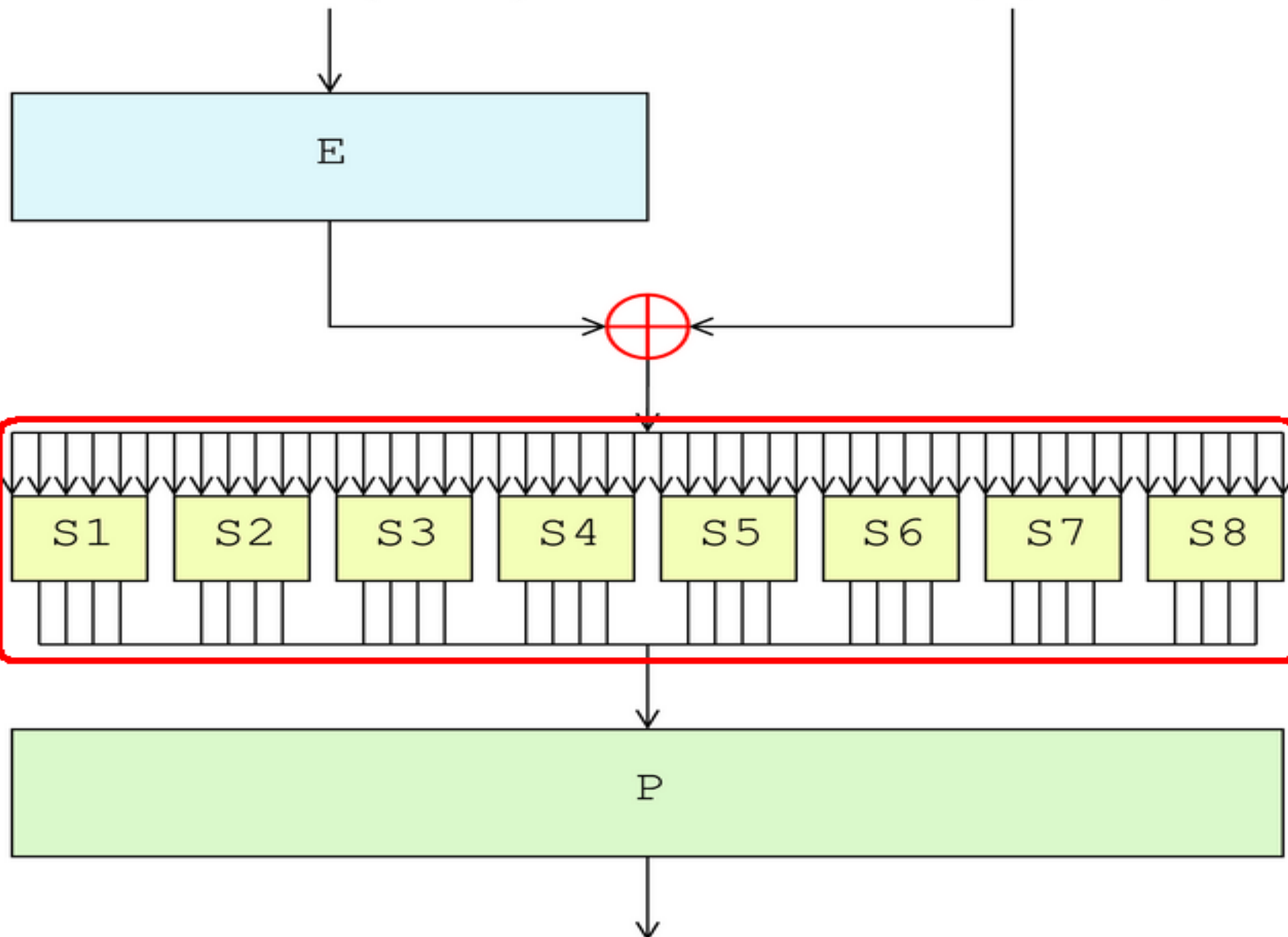
Vom calcula

$S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)$, unde $Si(Bi)$ este rezultatul folosirii permutației Si .

Pentru a nu uita, fiecare funcție S de va avea drept intrare un grup de 6 biți și va avea drept ieșire un grup de 4 biți.

Half Block (32 bits)

Subkey (48 bits)



		S_1															
		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

- a. Primul și ultimul bit din cei 6 va reprezenta un număr binar de 2 cifre din intervalul $[0,3]$ și îl vom numi generic **I**.
- b. Cei 4 biți din mijlocul blocului de 6 biți reprezintă un număr di intervalul $[0,15]$ și îl vom numi generic **J**.
- c. Vom căuta numărul aflat pe linia I și coloana J în matricea de mai sus. Acesta va fi ieșirea funcției $S_1(B)$.

Ex:

$$B = 011011$$

$$I = 01 = 1 \text{ (linia)}$$

$$J = 1101 = 13 \text{ (coloana)}$$

$$S_1(011011) = 5 \text{ (0101)}$$

Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Tabelele Si folosite în mod curent sunt:

S1

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

S2

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

S3

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

S4

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

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

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

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

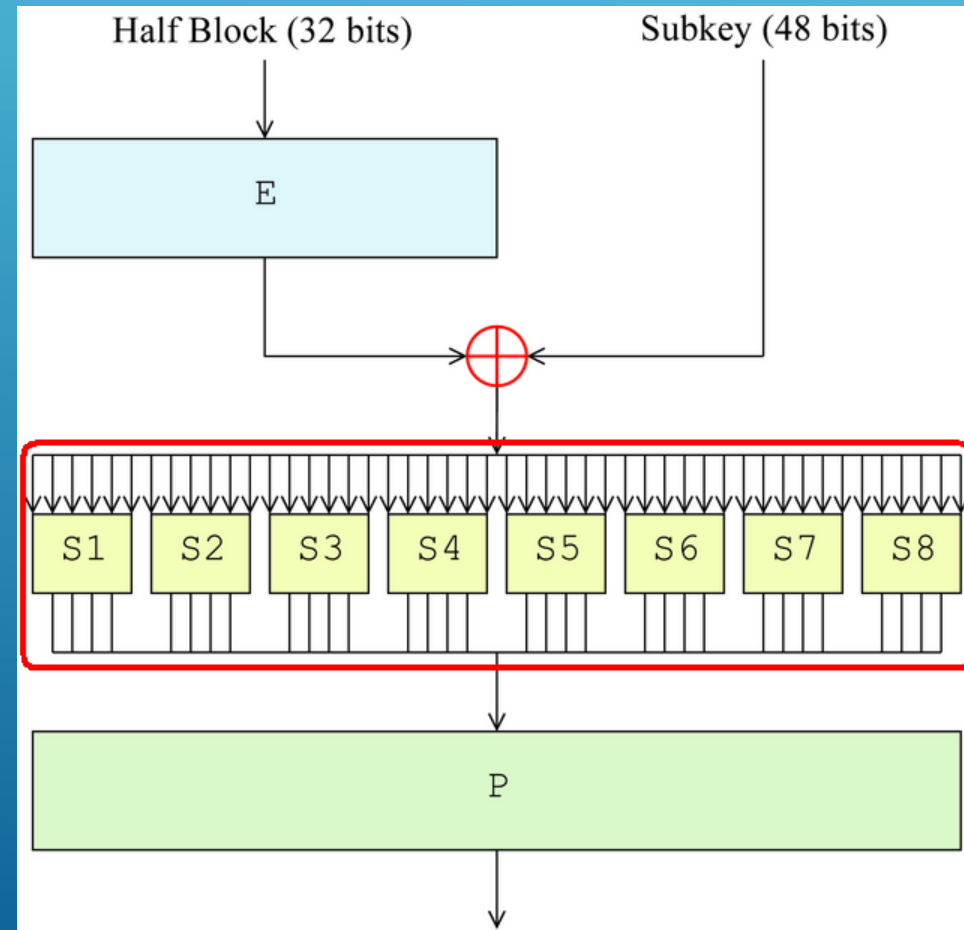
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Pasul final al calculării funcției f este efectuarea unei permutații f pentru $S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)$.
 $f = P(S1(B1)S2(B2)...S8(B8))$

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

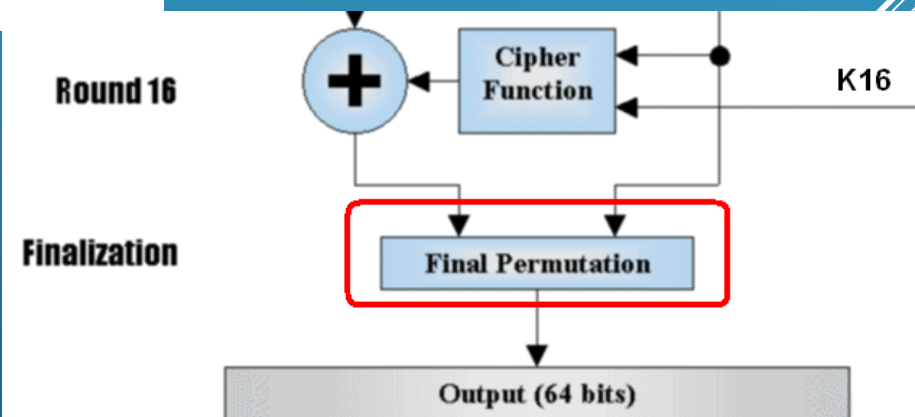


Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

După cele 16 iterații, vom obține L16 și R16 pe care le vom concatena invers: R16L16 și vom aplica o ultimă permutație IP-1, după cum urmează:

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

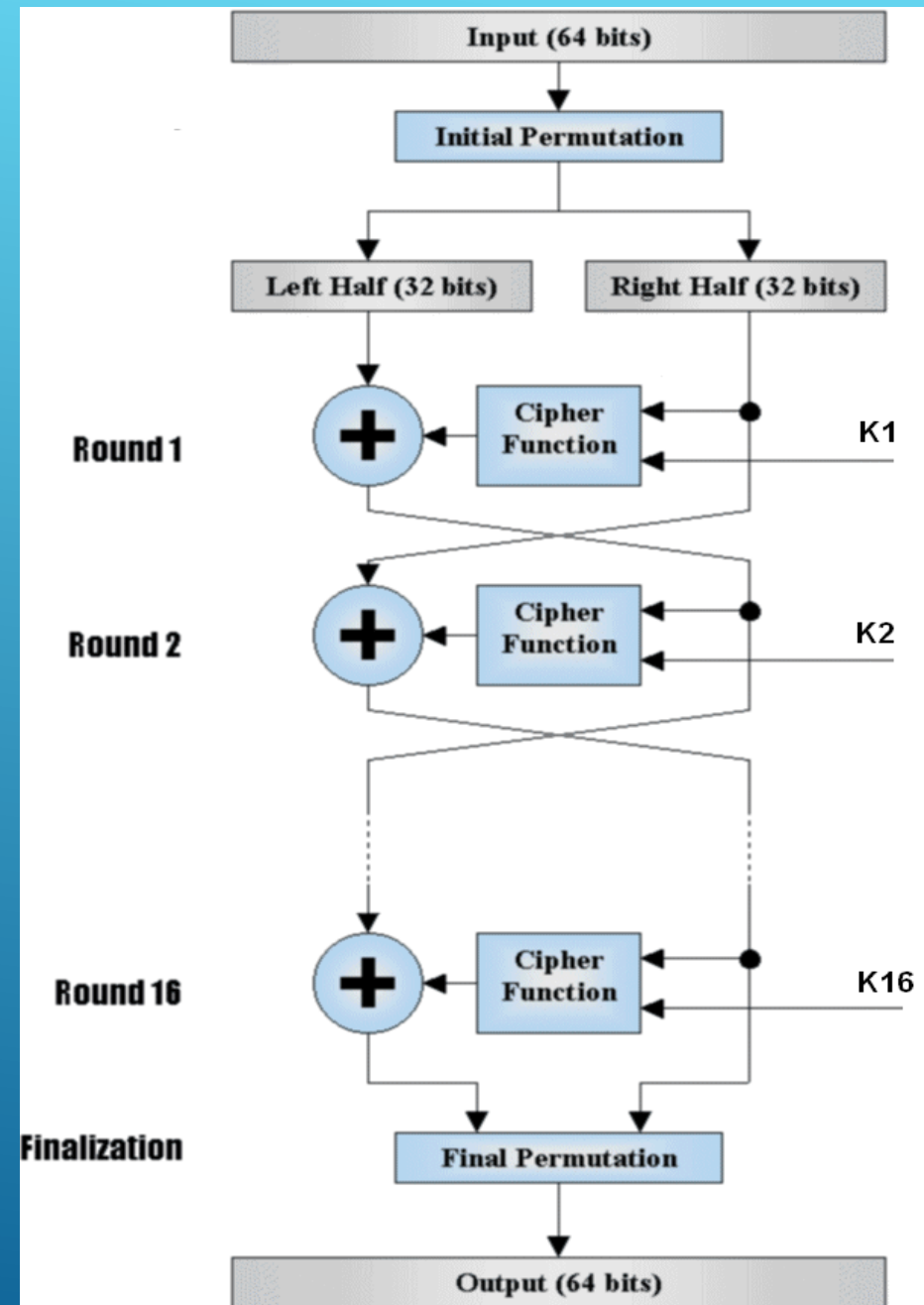


Algoritmul de criptare DES (Data Encryption Standard)

3. DES, descriere în detaliu

Prelucrarea cheilor, funcții de implementat:

1. DES_Permut_IP();
2. DES_Divide_L0_R0() ;
3. DES_Permut_E() ;
4. DES_Permut_Si ();
5. DES_Permut_P() (sau DES_Compute_F());
6. DES_Compute_Li_Ri());
7. DES_Compute_Inv_IP());
8. DES_Encrypt();



Algoritmul de criptare DES (Data Encryption Standard)

4. Moduri de aplicare a algoritmului DES

Deoarece algoritmul DES se aplică asupra unor blocuri de 64 de biți este denumit generic **algoritm de tip ECB (*Electronic Code Book*)**.

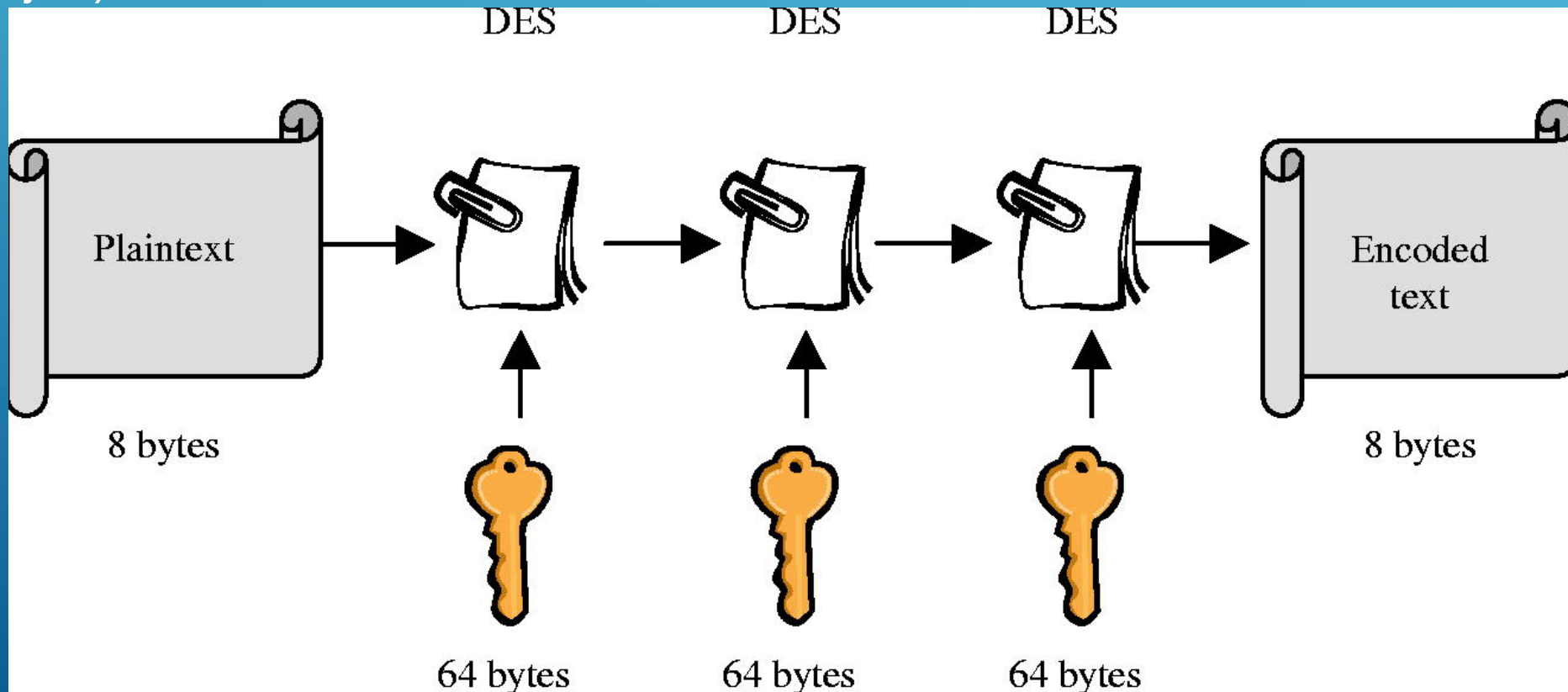
Există de asemenea două variante ale DES: **CBC (*Chain Block Coding*)** și **CFB (*Cipher Feedback*)** în care fiecare pachet depinde într-un mod sau altul de pachetul (pachetele) anterior criptat printr-o operație de tip XOR.

Deoarece este un algoritm ECB, putem imagina algoritmi ce funcționează pe 128 sau 256 de biți ce folosesc într-un mod sau altul criptarea sau decriptarea DES. Una dintre aceste modalități de criptare des folosită în industrie este și Triple DES.

Algoritmul de criptare DES (Data Encryption Standard)

5. Triple-DES

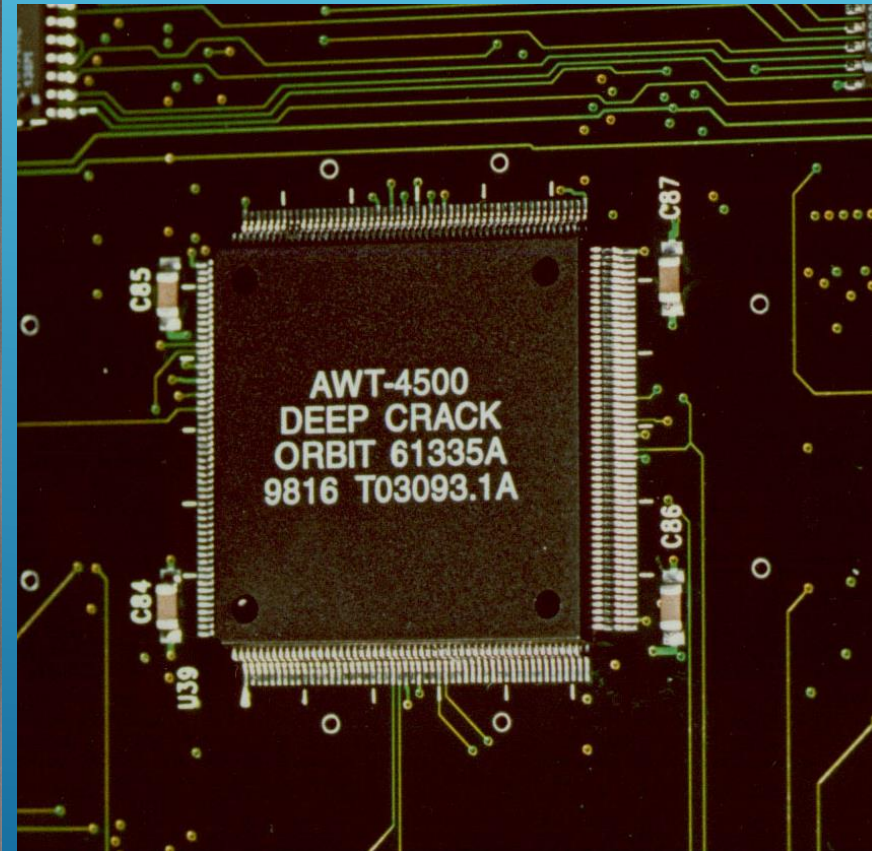
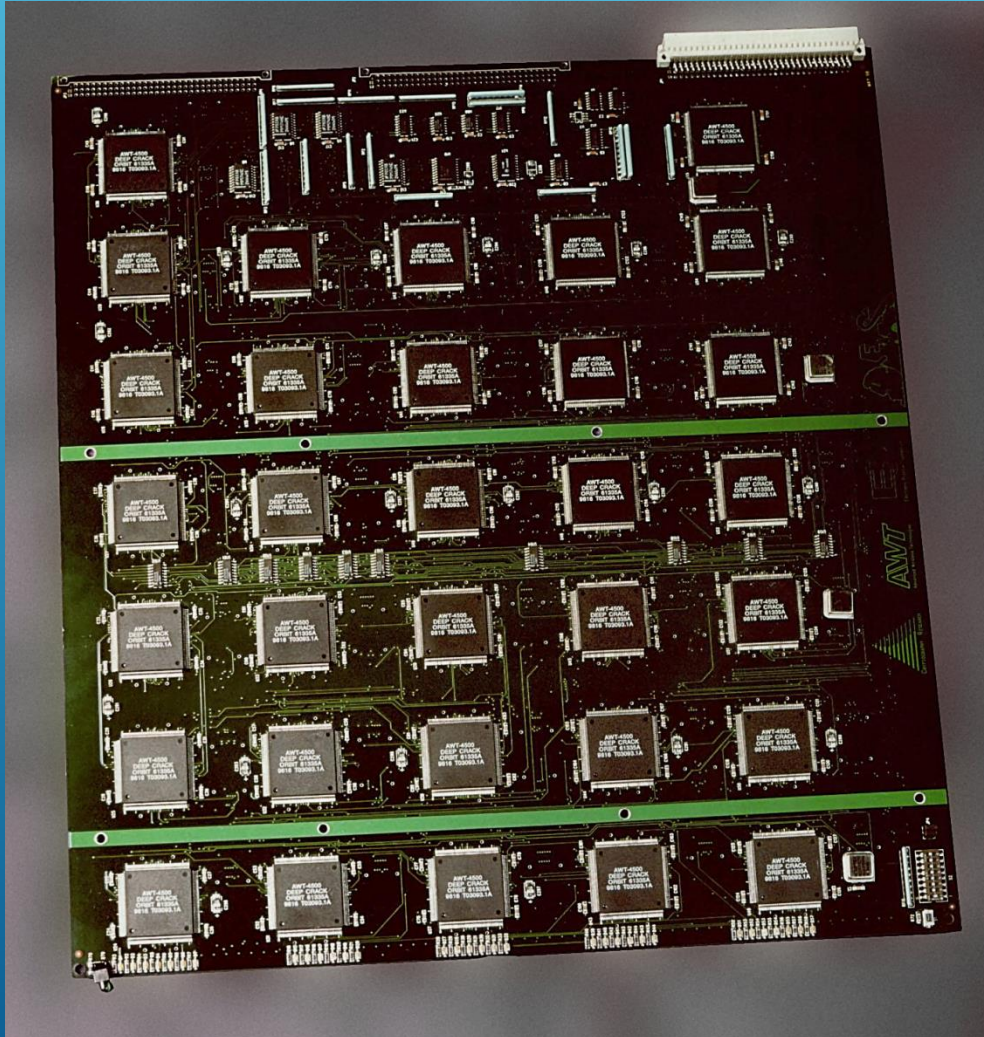
În ultimii ani, pentru a crește securitatea algoritmului DES, se folosește o altă variantă a sa denumită generic Triple-DES ce folosește două chei, după cum urmează (vezi figura de mai jos):



Algoritmul de criptare DES (Data Encryption Standard)

6. Mașini de căutare a chilor

6.1 EFF DES cracker ("Deep Crack")



Algoritmul de criptare DES (Data Encryption Standard)

6. Mașini de căutare a chilor

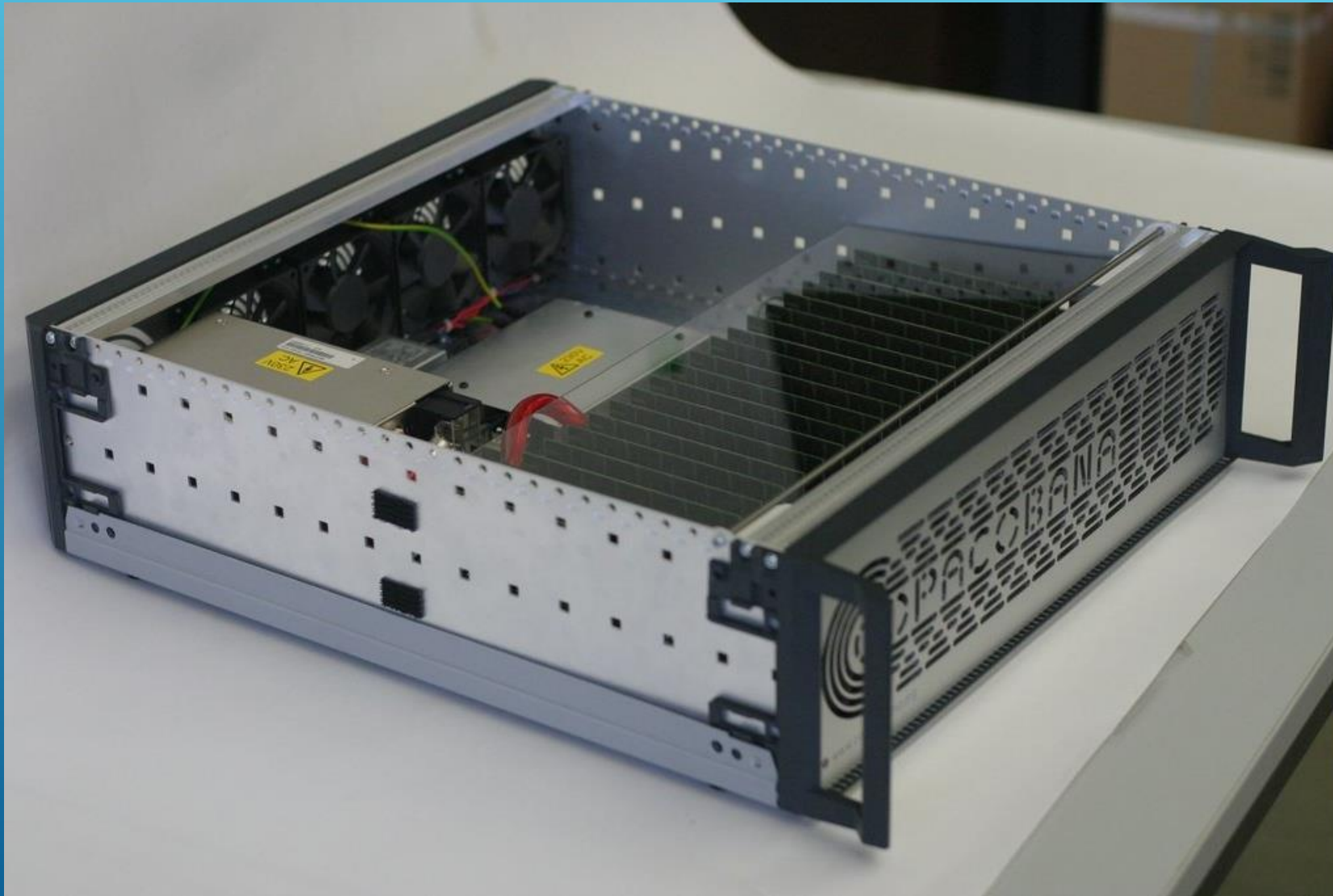
6.1 COPACOBANA (University of Bochum and Kiel, Germany)



Algoritmul de criptare DES (Data Encryption Standard)

6. Mașini de căutare a chilor

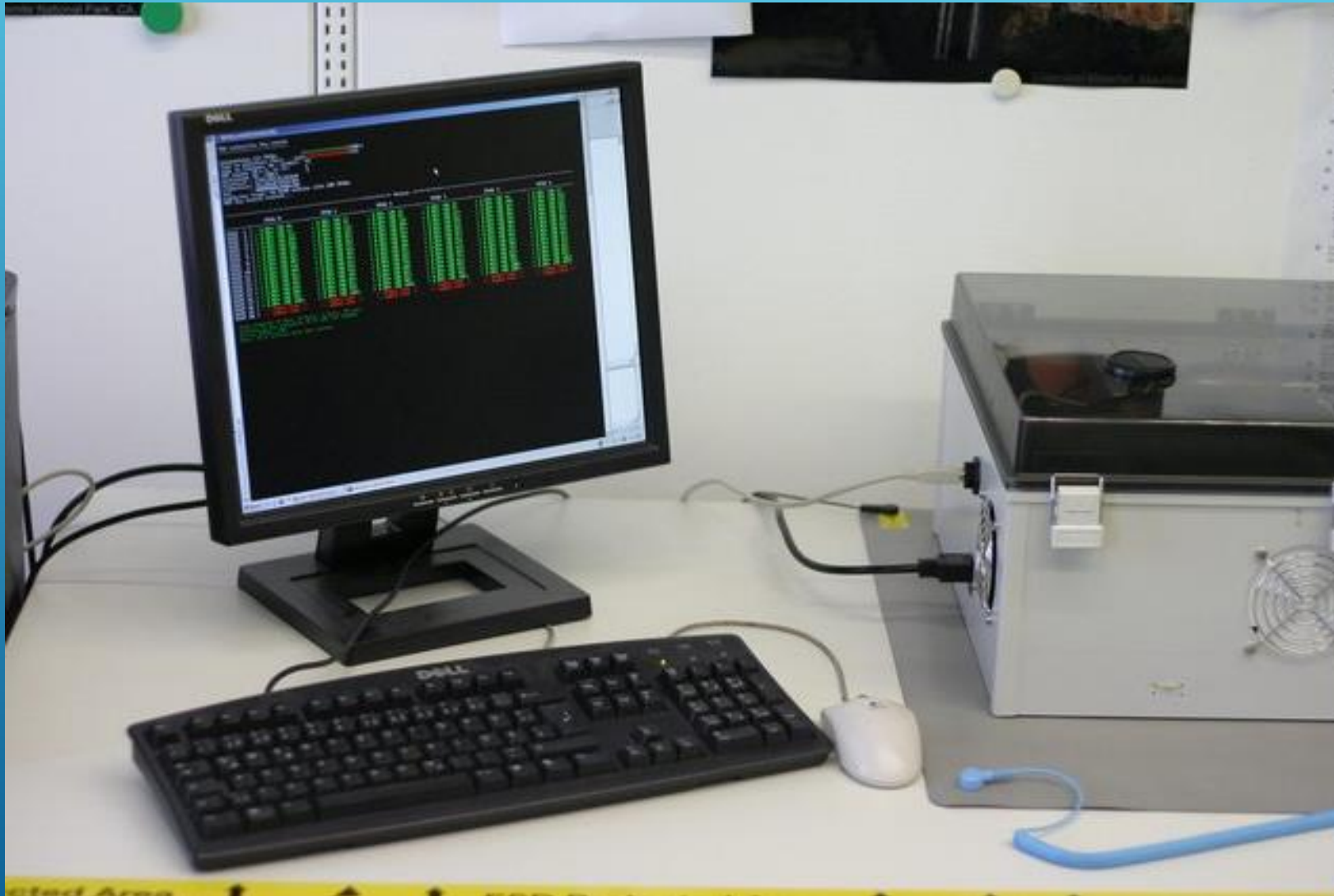
6.1 COPACOBANA (University of Bochum and Kiel, Germany)



Algoritmul de criptare DES (Data Encryption Standard)

6. Mașini de căutare a chilor

6.1 COPACOBANA (University of Bochum and Kiel, Germany)



Algoritmul de criptare DES (Data Encryption Standard)

6. Mașini de căutare a chilor

6.1 COPACOBANA (University of Bochum and Kiel, Germany)

```
DES key search (version 0.6)

DES exhaustive key search
-----
programming all FPGAs: >-----<108>
type in plaintext (hex)  nsh>1341343141343444<lsh
type in subspace (0...16383): 55
DES engine (0,1,2, or 3): 1
percentage (0...100): 3
plaintext: 1341343141343444
ciphertext: 36ca864f9359f49b
key: 0000000000000000
expansion stage: 18 DIMM modules with 108 FPGAs
DES key search started...

----- Status -----

```

	FPGA 0	FPGA 1	FPGA 2	FPGA 3	FPGA 4	FPGA 5
SLOT 1	✓ 2.73x (KS 0)	✓ 2.73x (KS 1)	✓ 2.73x (KS 2)	✓ 2.73x (KS 3)	✓ 2.73x (KS 4)	✓ 2.73x (KS 5)
SLOT 2	✓ 2.73x (KS 6)	✓ 2.73x (KS 7)	✓ 2.73x (KS 8)	✓ 2.73x (KS 9)	✓ 2.73x (KS 10)	✓ 2.73x (KS 11)
SLOT 3	✓ 2.73x (KS 12)	✓ 2.73x (KS 13)	✓ 2.73x (KS 14)	✓ 2.73x (KS 15)	✓ 2.73x (KS 16)	✓ 2.73x (KS 17)
SLOT 4	✓ 2.73x (KS 18)	✓ 2.73x (KS 19)	✓ 2.73x (KS 20)	✓ 2.73x (KS 21)	✓ 2.73x (KS 22)	✓ 2.73x (KS 23)
SLOT 5	✓ 2.73x (KS 24)	✓ 2.73x (KS 25)	✓ 2.73x (KS 26)	✓ 2.73x (KS 27)	✓ 2.73x (KS 28)	✓ 2.73x (KS 29)
SLOT 6	✓ 2.73x (KS 30)	✓ 2.73x (KS 31)	✓ 2.73x (KS 32)	✓ 2.73x (KS 33)	✓ 2.73x (KS 34)	✓ 2.73x (KS 35)
SLOT 7	✓ 2.73x (KS 36)	✓ 2.73x (KS 37)	✓ 2.73x (KS 38)	✓ 2.73x (KS 39)	✓ 2.73x (KS 40)	✓ 2.73x (KS 41)
SLOT 8	✓ 2.73x (KS 42)	✓ 2.73x (KS 43)	✓ 2.73x (KS 44)	✓ 2.73x (KS 45)	✓ 2.73x (KS 46)	✓ 2.73x (KS 47)
SLOT 9	✓ 2.73x (KS 48)	✓ 2.73x (KS 49)	✓ 2.74x (KS 50)	✓ 2.74x (KS 51)	✓ 2.74x (KS 52)	✓ 2.74x (KS 53)
SLOT 10	✓ 2.74x (KS 54)	↑: SUCCESSFUL !!	✓ 2.74x (KS 56)	✓ 2.74x (KS 57)	✓ 2.74x (KS 58)	✓ 2.74x (KS 59)
SLOT 11	✓ 2.74x (KS 60)	✓ 2.74x (KS 61)	✓ 2.74x (KS 62)	✓ 2.74x (KS 63)	✓ 2.74x (KS 64)	✓ 2.74x (KS 65)
SLOT 12	✓ 2.74x (KS 66)	✓ 2.74x (KS 67)	✓ 2.74x (KS 68)	✓ 2.74x (KS 69)	✓ 2.74x (KS 70)	✓ 2.74x (KS 71)
SLOT 13	✓ 2.74x (KS 72)	✓ 2.74x (KS 73)	✓ 2.74x (KS 74)	✓ 2.74x (KS 75)	✓ 2.74x (KS 76)	✓ 2.74x (KS 77)
SLOT 14	✓ 2.74x (KS 78)	✓ 2.74x (KS 79)	✓ 2.74x (KS 80)	✓ 2.74x (KS 81)	✓ 2.74x (KS 82)	✓ 2.74x (KS 83)
SLOT 15	✓ 2.74x (KS 84)	✓ 2.74x (KS 85)	✓ 2.74x (KS 86)	✓ 2.74x (KS 87)	✓ 2.74x (KS 88)	✓ 2.74x (KS 89)
SLOT 16	✓ 2.74x (KS 90)	✓ 2.74x (KS 91)	✓ 2.74x (KS 92)	✓ 2.74x (KS 93)	✓ 2.74x (KS 94)	✓ 2.74x (KS 95)
SLOT 17	✓ 2.74x (KS 96)	✓ 2.74x (KS 97)	✓ 2.74x (KS 98)	✓ 2.74x (KS 99)	✓ 2.74x (KS 100)	✓ 2.74x (KS 101)
SLOT 18	✓ 2.74x (KS 102)	✓ 2.74x (KS 103)	✓ 2.74x (KS 104)	✓ 2.74x (KS 105)	✓ 2.74x (KS 106)	✓ 2.74x (KS 107)
SLOT 19	--- empty slot ---	--- empty slot ---	--- empty slot ---	--- empty slot ---	--- empty slot ---	--- empty slot ---
SLOT 20	--- empty slot ---	--- empty slot ---	--- empty slot ---	--- empty slot ---	--- empty slot ---	--- empty slot ---

```
Time elapsed: 0 days, 0 hours, 5 mins, 0 secs
Working in key subspaces 0 to 107 (of 16384)
Active FPGAs: 108
Rate: 43.2 billion keys per second

>>> Key found in module 10, FPGA 1 at internal value [2e00000000000000]
>>> Verifying key interval: SUCCESSFUL => Key is lsb>0000000000000000<msb
>>> Key recovery in 388 seconds
```

Algoritmul de criptare DES (Data Encryption Standard)

6. Mașini de căutare a chilor

6.1 COPACOBANA (University of Bochum and Kiel, Germany)



Algoritmul de criptare DES (Data Encryption Standard)

7. Criptare DES cu C#

```
using System.Security.Cryptography;
using System.IO;

static byte[] bytes = ASCIIEncoding.ASCII.GetBytes("MyPassword");

public static string DESEncrypt(string originalString)
{
    if (String.IsNullOrEmpty(originalString))
    {
        throw new ArgumentNullException
            ("The string which needs to be encrypted can not be null.");
    }
    DESCryptoServiceProvider cryptoProvider = new DESCryptoServiceProvider();
    MemoryStream memoryStream = new MemoryStream();
    CryptoStream cryptoStream = new CryptoStream(memoryStream,
        cryptoProvider.CreateEncryptor(bytes, bytes), CryptoStreamMode.Write);
    StreamWriter writer = new StreamWriter(cryptoStream);
    writer.Write(originalString);
    writer.Flush();
    cryptoStream.FlushFinalBlock();
    writer.Flush();
    return Convert.ToBase64String(memoryStream.GetBuffer(), 0, (int)memoryStream.Length);
}
```