

Introducción a la Criptografía y a la Seguridad de la Información

Part 3

Data Encryption Standard

Jorge Camargo, PhD

Session 3

- **Data Encryption Standard DES**

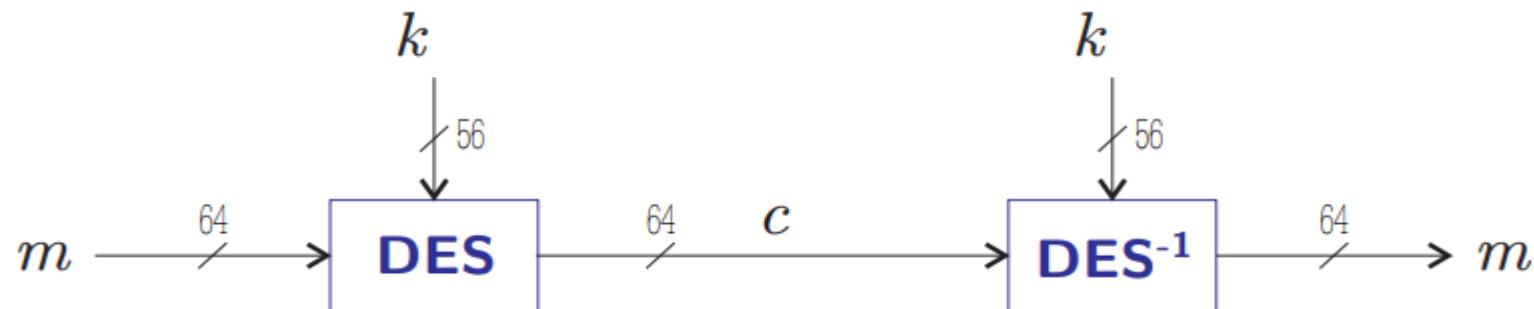
- Key Generator
- Steps of DES Algorithm
- PC-1
- PC-2
- IP
- Inner Function f
- E
- P
- S-Boxes
- IP^{-1}
- Example

Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a **block cipher** invented in the early 1970s by IBM and the U.S. government (US patent 3,962,539).

It operates on blocks of 64 bits using a secret key that is 56 bits long.

$$M = C = \{0, 1\}^{64}, K = \{0, 1\}^{56}.$$



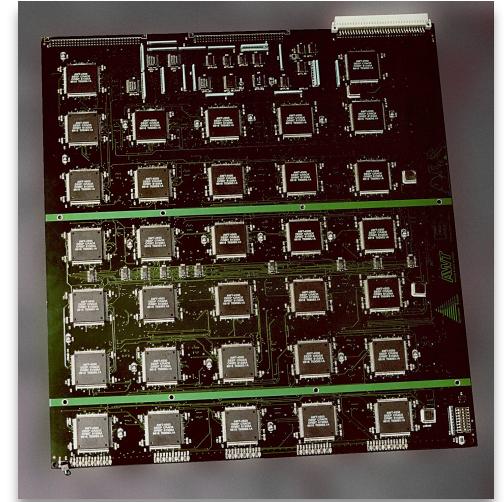
At the time it was believed that trying out all $72,057,594,037,927,936$ (72 quadrillion) possible keys (a seven with 16 zeros) would be impossible because computers could not possibly ever become fast enough.

DES has been replaced by AES as a standard. We will use DES to illustrate the principles of modern symmetric ciphers

Steps of DES Algorithm

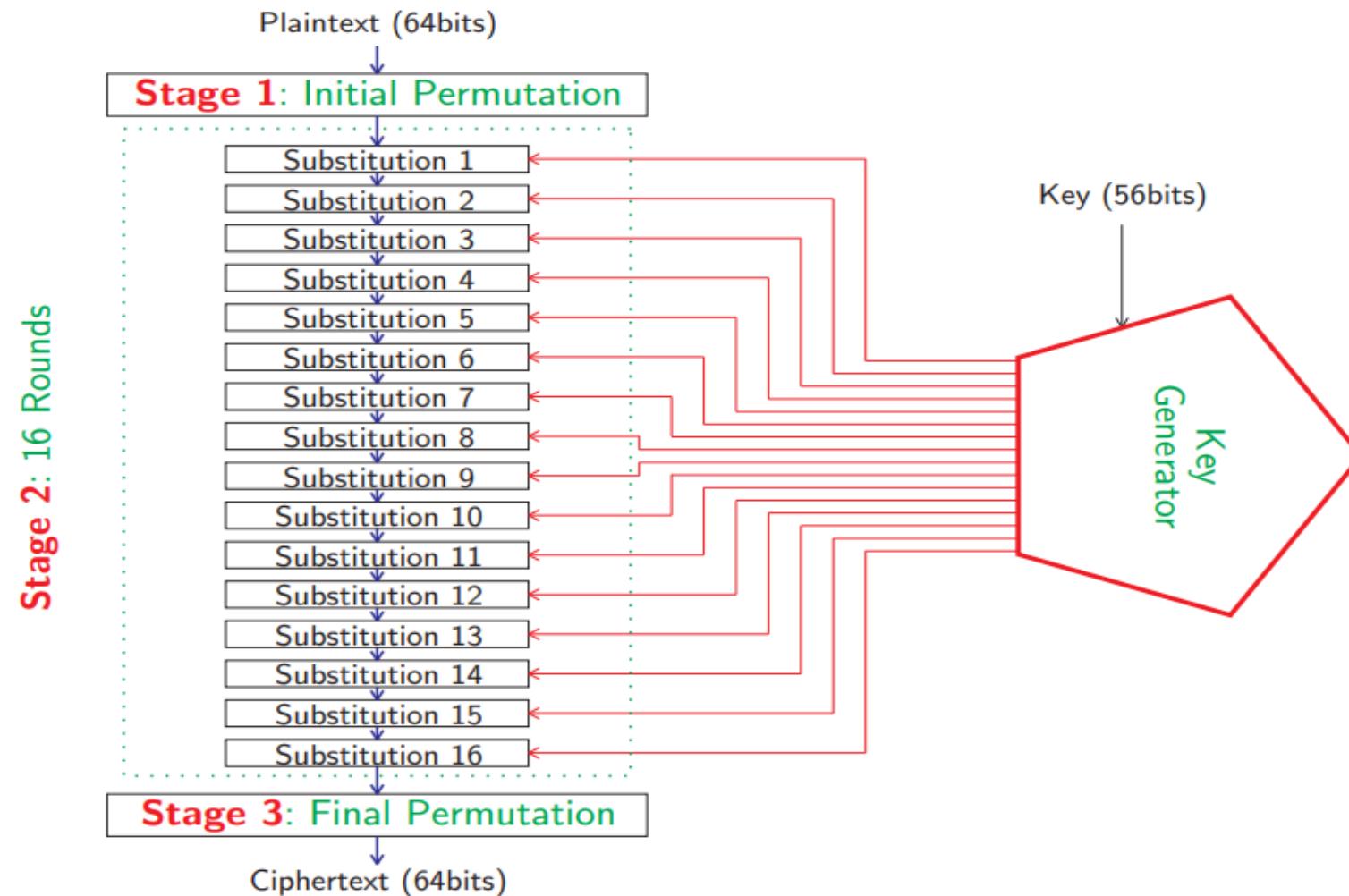
The algorithm has 3 stages (18 steps):

- **Stage 1:** Initial Permutation
- **Stage 2:** 16 Operations (rounds)
- **Stage 3:** Final Permutation



In 1998, the Electronic Frontier Foundation (EFF) built a machine that could crack the DES algorithm by brute-force; called DES Deep Crack, it could find a 56-bit DES key in an average of 4.5 days.

Steps of DES Algorithm (cont.)



Key Generator

DES must first create 16 subkeys as follows:

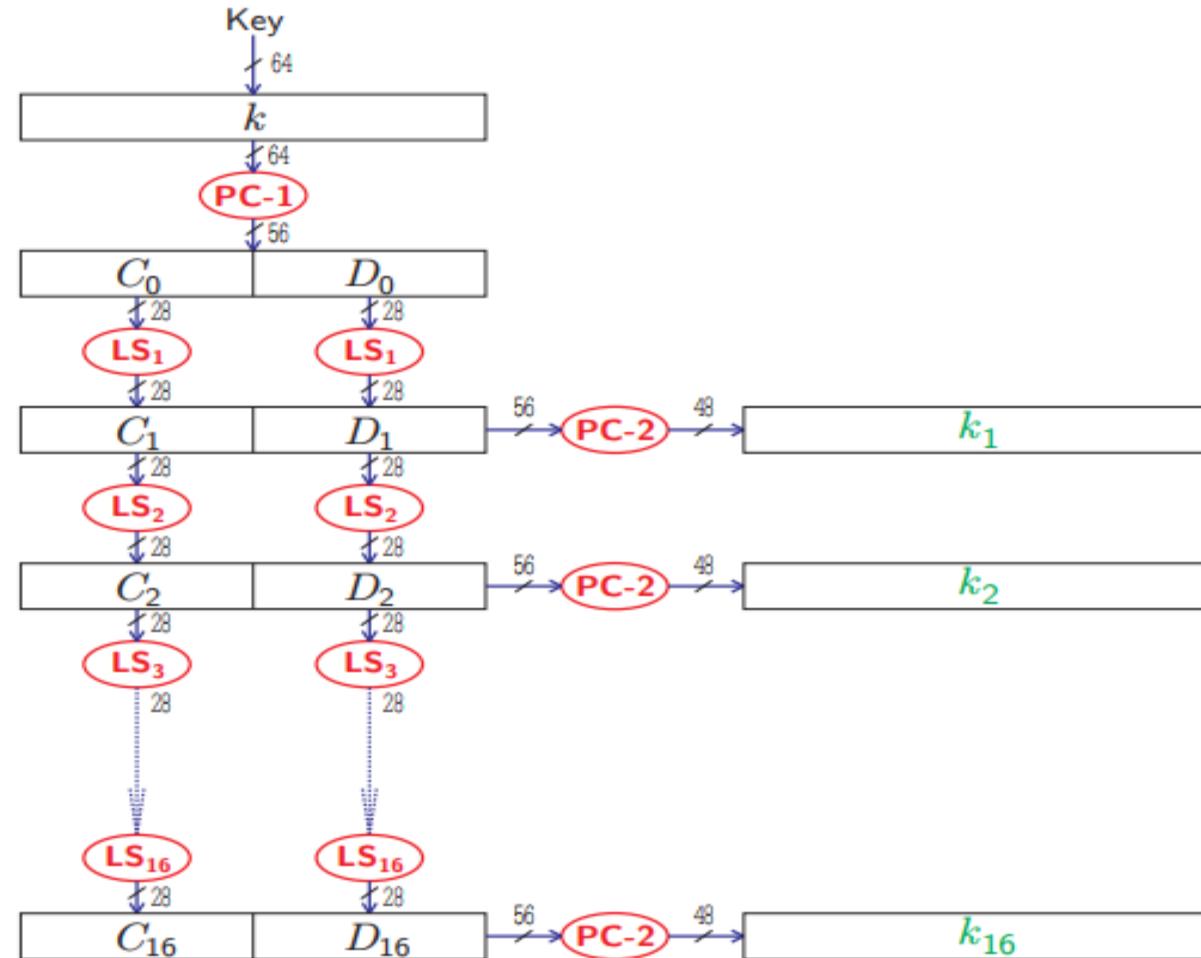
1. From a given bitstring key k of length 64, of which 56 bits comprise the key and 8 bits are parity-check bits(for error-detection), compute $k_0 = \text{PC-1}(k) = C_0D_0$, where C_0 comprise the first 28 bits of $\text{PC-1}(k)$ and D_0 the last 28 bits.
2. For i ranging 1 to 16, compute:

$$C_i = LS_i(C_{i-1}), D_i = LS_i(D_{i-1}), k_i = \text{PC-2}(C_iD_i).$$

LS_i represents a cyclic shift (to the left) of either one or two positions, depending on the value of i : by 1 if $i = 1, 2, 9, 16$, by 2 otherwise. PC-2 is another fixed permutation.

The bits in positions 8,16,24,32,40,48,56 and 64 of k are defined so that each byte contains an odd number of 1's. Hence, a single error can be detected within each group of 8 bits. The parity-check bits are ignored in the computation of the key.

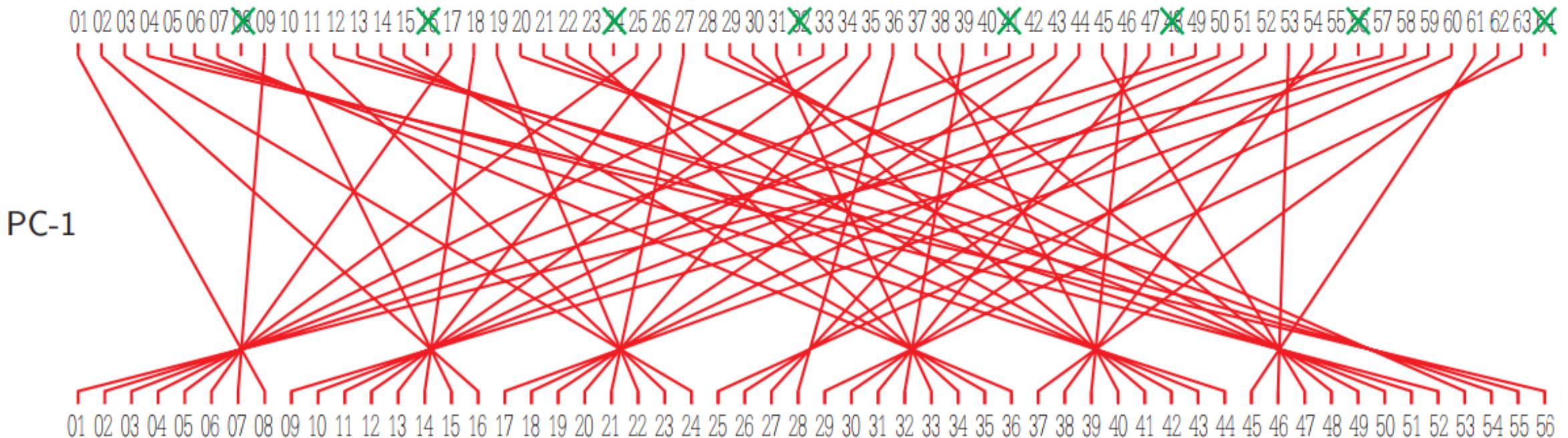
Computation of the 16 DES-Subkeys



Permuted Choice 1 (PC-1)



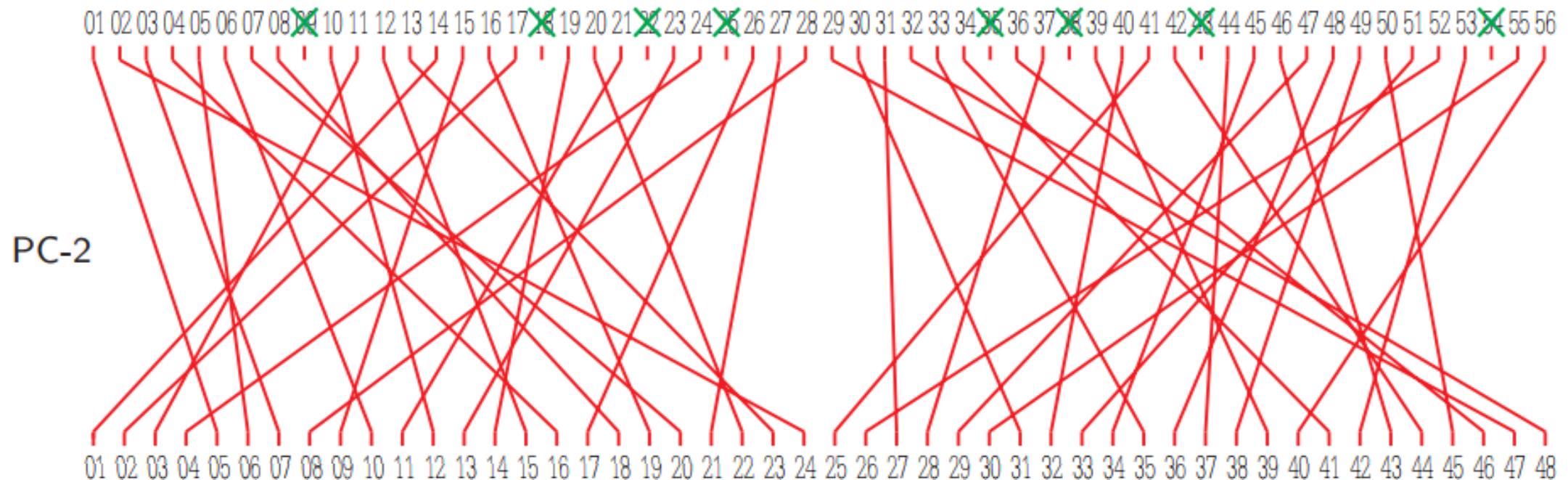
bit	01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
PC-1	57 49 41 33 25 17 09 01 58 50 42 34 26 18 10 02 59 51 43 35 27 19 11 03 60 52 44 36 63 55 47 39 31 23 15 07 62 54 46 38 30 22 14 06 61 53 45 37 29 21 13 05 28 20 12 04



Permuted Choice 2 (PC-2)



bit	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
PC-2	14	17	11	24	01	05	03	28	15	06	21	10	23	19	12	04	26	08	16	07	27	20	13	02	41	52	31	37	47	55	30	40	51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32



Stage 1: Initial Permutation

Given a plaintext m , a bitstring m_0 is constructed by permuting the bits of m according to a (fixed) **initial permutation IP**.

We write $m_0 = IP(x) = L_0R_0$, where L_0 comprises the first 32 bits of m_0 and R_0 the last 32 bits.

If the block is shorter than 64 bits, it should be padded with zeros.

Initial Permutation (IP)



IP

$$(01\ 02\ 03\ 04\ 05\ 06\ 07\ 08\ 09\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21\ 22\ 23\ 24\ 25\ 26\ 27\ 28\ 29\ 30\ 31\ 32\ 33\ 34\ 35\ 36\ 37\ 38\ 39\ 40\ 41\ 42\ 43\ 44\ 45\ 46\ 47\ 48\ 49\ 50\ 51\ 52\ 53\ 54\ 55\ 56\ 57\ 58\ 59\ 60\ 61\ 62\ 63\ 64)$$
$$(58\ 50\ 42\ 34\ 26\ 18\ 10\ 02\ 60\ 52\ 44\ 36\ 28\ 20\ 12\ 04\ 62\ 54\ 46\ 38\ 30\ 22\ 14\ 06\ 64\ 56\ 48\ 40\ 32\ 24\ 16\ 08\ 57\ 49\ 41\ 33\ 25\ 17\ 09\ 01\ 59\ 51\ 43\ 35\ 27\ 19\ 11\ 03\ 61\ 53\ 45\ 37\ 29\ 21\ 13\ 05\ 63\ 55\ 47\ 39\ 31\ 23\ 15\ 07)$$

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 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

IP

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 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

IP^{-1}

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 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

IP^{-1}

$$(01\ 02\ 03\ 04\ 05\ 06\ 07\ 08\ 09\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21\ 22\ 23\ 24\ 25\ 26\ 27\ 28\ 29\ 30\ 31\ 32\ 33\ 34\ 35\ 36\ 37\ 38\ 39\ 40\ 41\ 42\ 43\ 44\ 45\ 46\ 47\ 48\ 49\ 50\ 51\ 52\ 53\ 54\ 55\ 56\ 57\ 58\ 59\ 60\ 61\ 62\ 63\ 64)$$
$$(40\ 08\ 48\ 16\ 56\ 24\ 64\ 32\ 39\ 07\ 47\ 15\ 55\ 23\ 63\ 31\ 38\ 06\ 46\ 14\ 54\ 22\ 62\ 30\ 37\ 05\ 45\ 13\ 53\ 21\ 61\ 29\ 36\ 04\ 44\ 12\ 52\ 20\ 60\ 28\ 35\ 03\ 43\ 11\ 51\ 19\ 59\ 27\ 34\ 02\ 42\ 10\ 50\ 18\ 58\ 26\ 33\ 01\ 41\ 09\ 49\ 17\ 57\ 25)$$

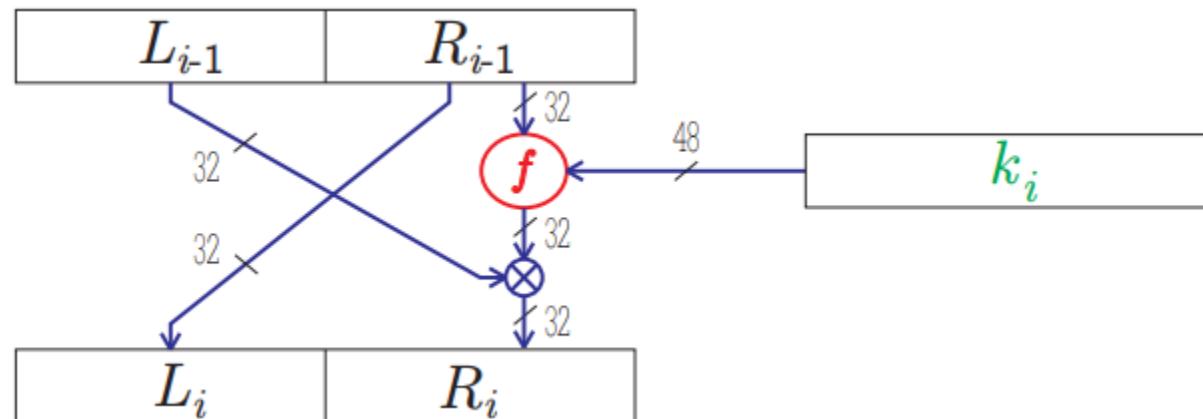
Stage 2: 16 Rounds

16 iterations of a certain function are computed.

We compute $L_i R_i$, for $1 \leq i \leq 16$, according to the following rule:

$$L_i = R_{i-1}, R_i = L_{i-1} \otimes f(R_{i-1}, k_i),$$

where \otimes denotes the exclusive-or, f is the inner function of DES and it's described later, and k_1, k_2, \dots, k_{16} are the subkeys we already computed.

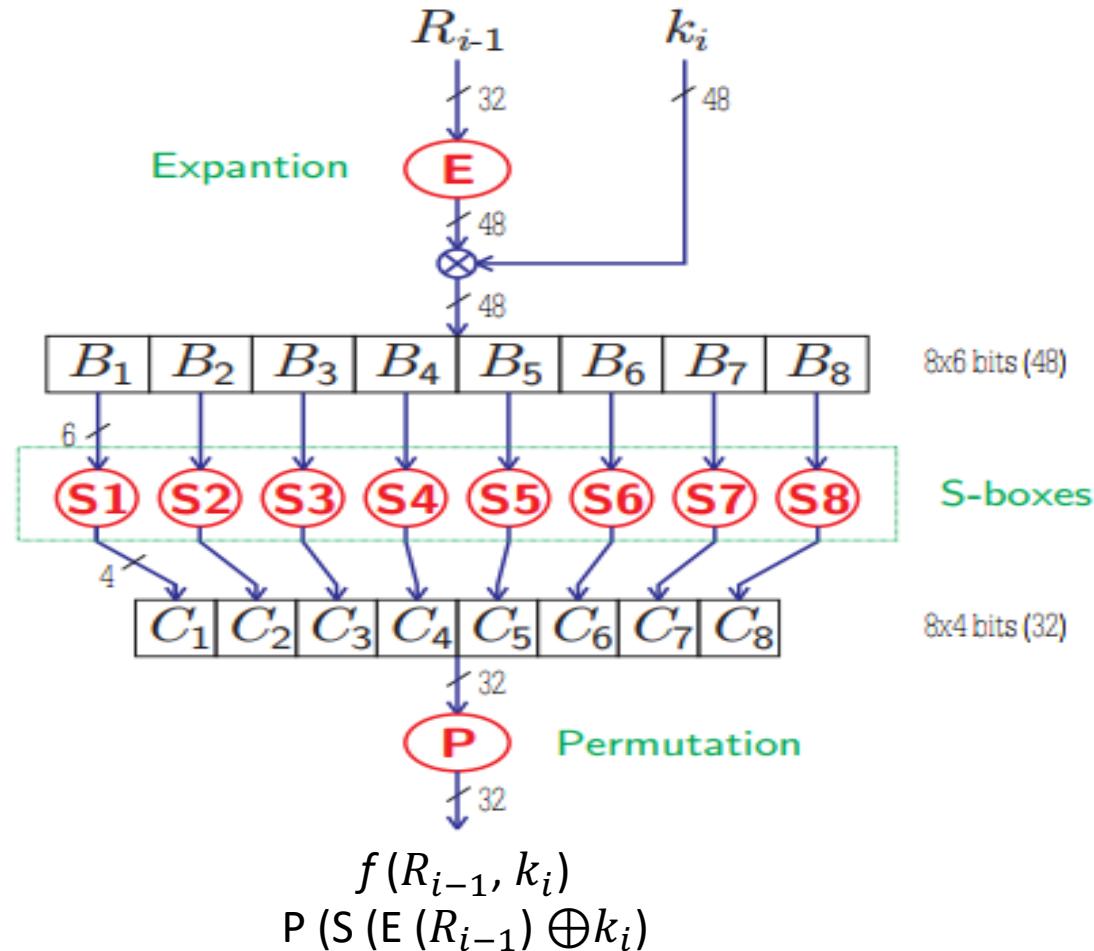


Inner Function f

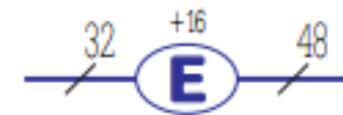
The function f takes as input a first argument R_{i-1} of length 32, and a second argument k_i of length 48, and produces as output a bitstring of length 32. The following steps are executed:

1. The first argument R_{i-1} is expanded to a bitstring of length 48 according to a fixed expansion function E . $E(R_{i-1})$ consists of 32 bits from R_{i-1} , permuted in a certain way, with 16 of the bits appearing twice.
2. Compute $R_{i-1} \otimes k_i$ and write the result as the concatenation of eight 6-bit strings $B = B_1B_2B_3B_4B_5B_6B_7B_8$.
3. Apply $s_j(B_j)$ to every block B_j . S_j is a fixed 4×16 array whose entries come from the integers 0-15. For each $B_j = b_1b_2b_3b_4b_5b_6$, the two bits b_1b_6 determine the binary representation of the row r of $S_j(r, c)$ and the four bits $b_2b_3b_4b_5$ determines the binary representation of the column. $C_j = s_j(B_j)$ is defined to be the entry $s_j(r, c)$, written in binary as a bitstring of length four.
4. The bitstring $C = C_1C_2C_3C_4C_5C_6C_7C_8$ of length 32 is permuted according to a fixed permutation P . $f(R_{i-1}, k_i) = P(C)$.

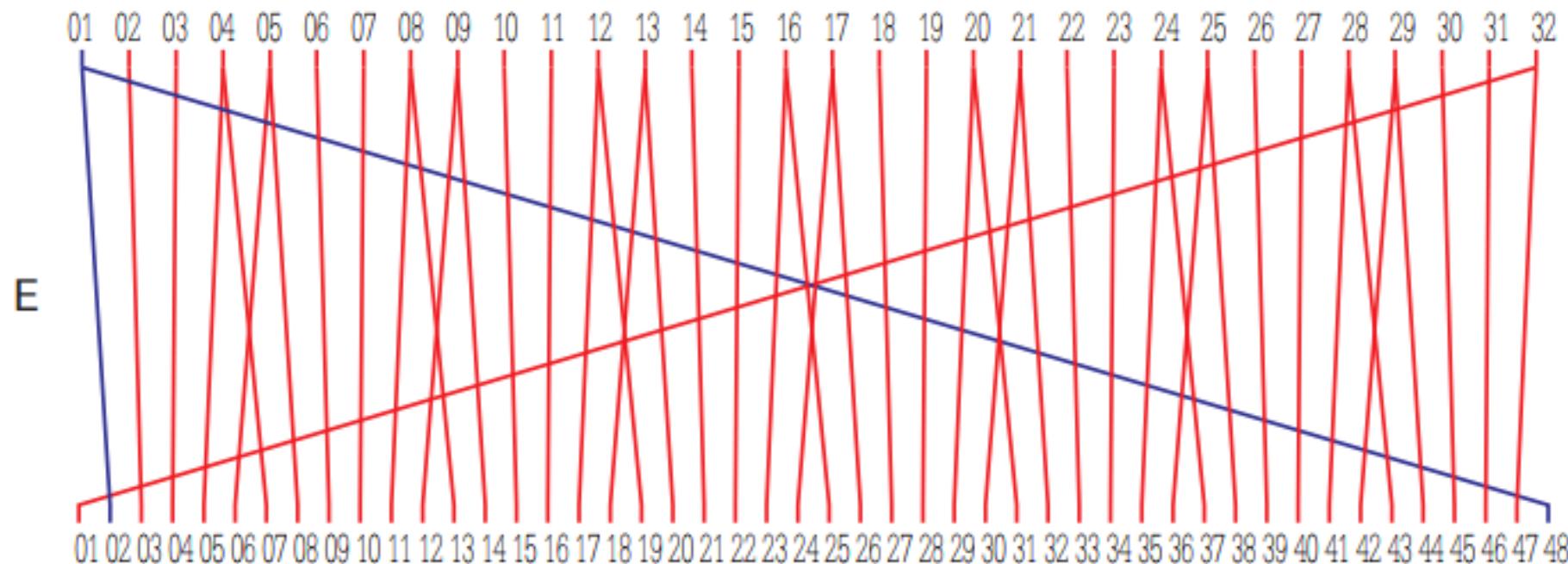
Inner Function f



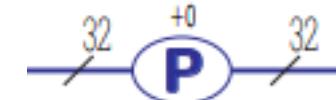
Expansion (E)



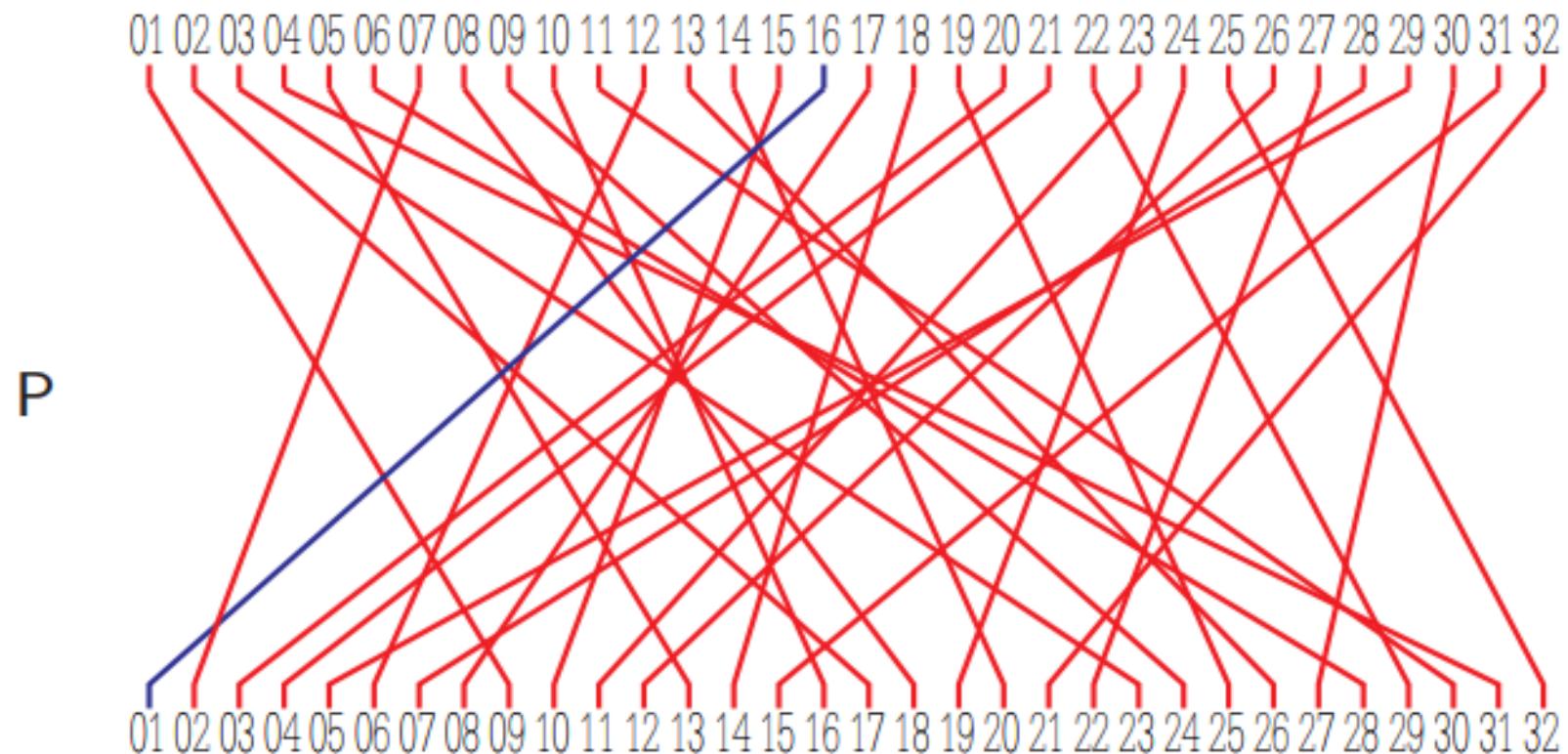
bit	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
E	32	01	02	03	04	05	06	07	08	09	08	09	10	11	12	13	12	13	14	15	16	17	16	17	18	19	20	21	20	21	22	23	24	25	24	25	26	27	28	29	28	29	30	31	32	01		



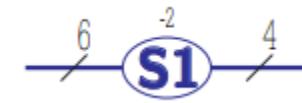
Permutation (P)



$P \begin{pmatrix} 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \\ 16 & 07 & 20 & 21 & 29 & 12 & 28 & 17 & 01 & 15 & 23 & 26 & 05 & 18 & 31 & 10 & 02 & 08 & 24 & 14 & 32 & 27 & 03 & 09 & 19 & 13 & 30 & 06 & 22 & 11 & 04 & 25 \end{pmatrix}$



S-Box 1 (S1)



		<i>c</i>															
S1		00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
<i>r</i>	00	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
	01	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
	02	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
	03	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

$$B_j = b_1 b_2 b_3 b_4 b_5 b_6$$

$$r = b_1 b_6 \quad [0..3]$$

$$c = b_2 b_3 b_4 b_5 \quad [0..15]$$

Example: Compute S1(011000)

$$S1(011000) = S1(00, 1100) = S1(0, 12) = 5 = 0101$$

$$S1(011000) = 0101$$

S-Boxes 1–8 (S1,S2,S3,S4,S5,S6,S7,S8)

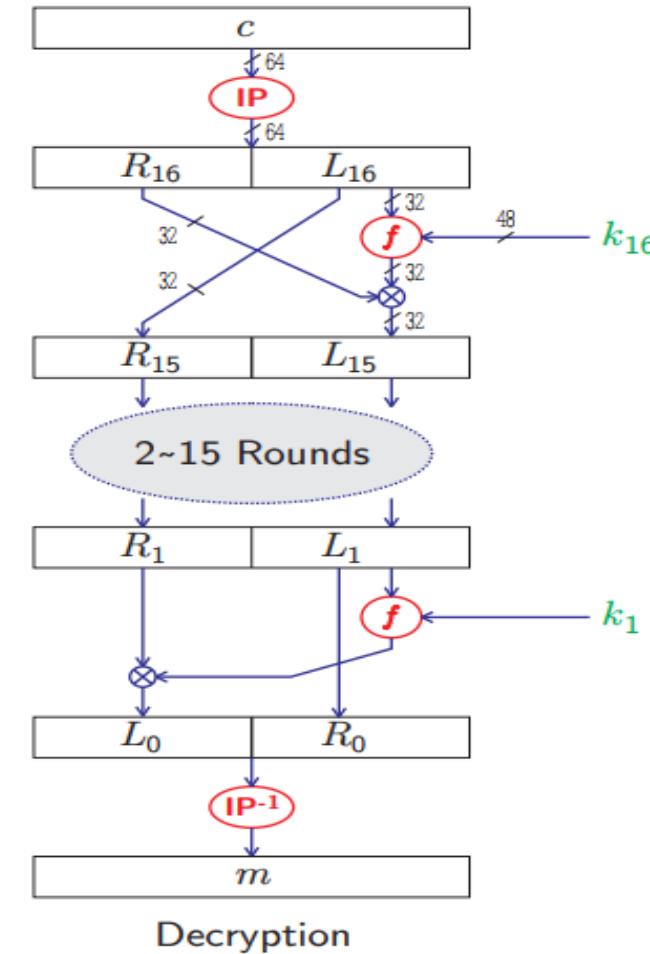
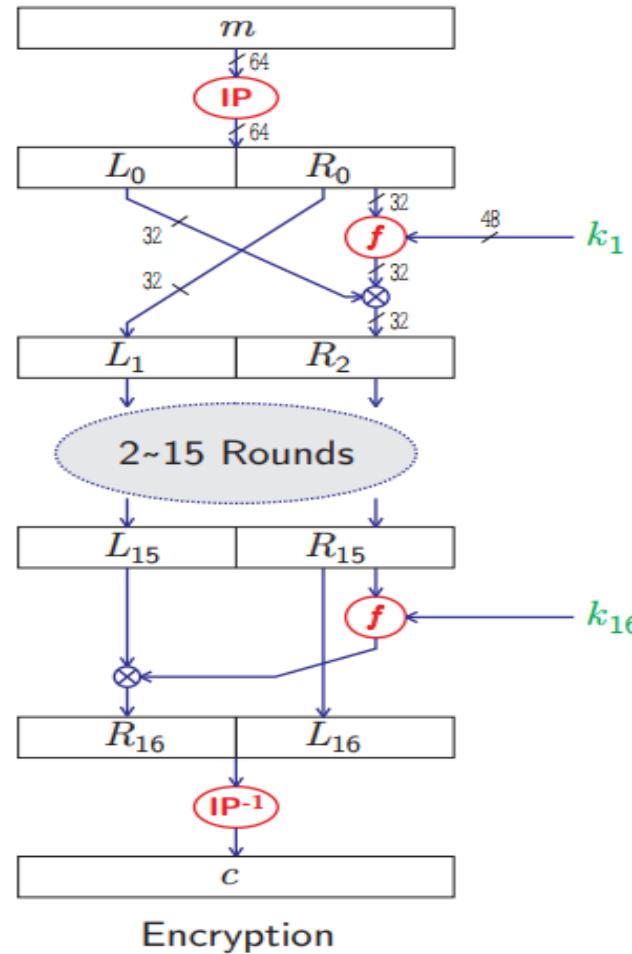
		c																		
		c																		
		c																		
		00	00	01	02	03	04	05	06	07	08	09	00	10	11	12	13	14	15	
<i>r</i>	S1	00	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07		
		01	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08		
		02	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00		
		03	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13		
<i>r</i>	S2	00	00	01	02	03	04	05	06	07	08	09	00	10	11	12	13	14	15	
		00	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10		
		01	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05		
		02	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15		
<i>r</i>	S3	00	00	01	02	03	04	05	06	07	08	09	00	10	11	12	13	14	15	
		00	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08		
		01	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01		
		02	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07		
<i>r</i>	S4	00	00	01	02	03	04	05	06	07	08	09	00	10	11	12	13	14	15	
		00	07	13	14	03	00	06	09	10	01	02	08	05	11	12	04	15		
		01	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09		
		02	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04		
<i>r</i>		03	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14		
			00	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09	
			01	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06	
			02	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14	
			03	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03	
			00	00	01	02	03	04	05	06	07	08	09	00	10	11	12	13	14	15
			00	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11	
			01	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08	
			02	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06	
			03	04	03	02	12	09	05	15	10	11	14	01	07	06	00	08	13	
			00	00	01	02	03	04	05	06	07	08	09	00	10	11	12	13	14	15
			00	04	11	02	14	15	00	08	13	03	12	09	07	05	10	06	01	
			01	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06	
			02	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02	
			03	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12	
			00	00	01	02	03	04	05	06	07	08	09	00	10	11	12	13	14	15
			00	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07	
			01	01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02	
			02	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08	
			03	02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11	

Stage 1: Final Permutation

Apply the inverse permutation IP^{-1} to the bitstring $R_{16}L_{16}$. Obtain the ciphertext $c = IP^{-1}(R_{16}L_{16})$.

Note the inverted order of L₁₆ and R₁₆.

Encryption/Decryption



Example

Let $m = 0123456789ABCDEF$ and $k = 133457799BBCDFF1$, where m and k are in hexadecimal (base 16) format.

Part 1: Create 16 subkeys:

$k = 133457799BBCDFF1$.

This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc., and grouping together every eight bits, of which the last one in each group will be unused):

$k = 00010011\ 00110100\ 01010111\ 01111001\ 10011011\ 10111100\ 11011111\ 11110001$

Compute $k' = \text{PC-1}(k)$

$k' = 1111000\ 0110011\ 0010101\ 0101111\ 0101010\ 1011001\ 1001111\ 0001111$

Next, split this key into left and right halves, C_0 and D_0 , where each half has 28 bits.

$$C_0 = 1111000\ 0110011\ 0010101\ 0101111$$

$$D_0 = 0101010\ 1011001\ 1001111\ 0001111$$

Now, compute $C_i = LS_i(C_{i-1})$, $D_i = LS_i(D_{i-1})$ for $i = 1, 2, \dots, 16$

$$\begin{aligned} C_0 &= 1111000011001100101010101111 \\ C_1 &= 1110000110011001010101011111 \\ C_2 &= 1100001100110010101010111111 \\ C_3 &= 000011001100101010101111111 \\ C_4 &= 0011001100101010111111100 \\ C_5 &= 1100110010101011111110000 \\ C_6 &= 0011001010101111111000011 \\ C_7 &= 1100101010111111100001100 \\ C_8 &= 0010101011111110000110011 \\ C_9 &= 0101010111111110000110010 \\ C_{10} &= 0101010111111110000110011001 \\ C_{11} &= 010101111111000011001100101 \\ C_{12} &= 010111111100001100110010101 \\ C_{13} &= 0111111110000110011001010101 \\ C_{14} &= 1111111000011001100101010101 \\ C_{15} &= 1111100001100110010101010111 \\ C_{16} &= 1111000011001100101010101111 \end{aligned}$$

$$\begin{aligned} D_0 &= 0101010101100110011110001111 \\ D_1 &= 1010101011001100111100011110 \\ D_2 &= 0101010110011001111000111101 \\ D_3 &= 0101011001100111100011110101 \\ D_4 &= 0101100110011110001111010101 \\ D_5 &= 0110011001111000111101010101 \\ D_6 &= 1001100111100011110101010101 \\ D_7 &= 0110011110001111010101010110 \\ D_8 &= 1001111000111101010101011001 \\ D_9 &= 0011110001111010101010110011 \\ D_{10} &= 1111000111101010101011001100 \\ D_{11} &= 1100011110101010101100110011 \\ D_{12} &= 0001111010101010110011001111 \\ D_{13} &= 01111010101011001100111100 \\ D_{14} &= 1110101010101100110011110001 \\ D_{15} &= 1010101010110011001111000111 \\ D_{16} &= 0101010101100110011110001111 \end{aligned}$$

We now form the keys $k_i = \text{PC-2}(C_i D_i)$ for $i = 1, 2, 3, \dots, 16$.

$k_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$
 $k_2 = 011110\ 011010\ 111011\ 011001\ 110110\ 111100\ 100111\ 100101$
 $k_3 = 010101\ 011111\ 110010\ 001010\ 010000\ 101100\ 111110\ 011001$
 $k_4 = 011100\ 101010\ 110111\ 010110\ 110110\ 110011\ 010100\ 011101$
 $k_5 = 011111\ 001110\ 110000\ 000111\ 111010\ 110101\ 001110\ 101000$
 $k_6 = 011000\ 111010\ 010100\ 111110\ 010100\ 000111\ 101100\ 101111$
 $k_7 = 111011\ 001000\ 010010\ 110111\ 111101\ 100001\ 100010\ 111100$
 $k_8 = 111101\ 111000\ 101000\ 111010\ 110000\ 010011\ 101111\ 111011$
 $k_9 = 111000\ 001101\ 101111\ 101011\ 111011\ 011110\ 011110\ 000001$
 $k_{10} = 101100\ 011111\ 001101\ 000111\ 101110\ 100100\ 011001\ 001111$
 $k_{11} = 001000\ 010101\ 111111\ 010011\ 110111\ 101101\ 001110\ 000110$
 $k_{12} = 011101\ 010111\ 000111\ 110101\ 100101\ 000110\ 011111\ 101001$
 $k_{13} = 100101\ 111100\ 010111\ 010001\ 111110\ 101011\ 101001\ 000001$
 $k_{14} = 010111\ 110100\ 001110\ 110111\ 111100\ 101110\ 011100\ 111010$
 $k_{15} = 101111\ 111001\ 000110\ 001101\ 001111\ 010011\ 111100\ 001010$
 $k_{16} = 110010\ 110011\ 110110\ 001011\ 000011\ 100001\ 011111\ 110101$

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

Rewriting $m=0123456789ABCDEF$ in binary format, we get the 64-bit block of text:

$m = 000000010010001101000101011001111000100110101011100110111101111$

Compute $m' = \text{IP}(m)$

$m' = 11001100000000011001100111111111000010101010111000010101010$

Here the 58th bit of m is "1", which becomes the first bit of m' . The 50th bit of m is "1", which becomes the second bit of m' . The 7th bit of m is "0", which becomes the last bit of m' .

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

$L_0 = 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111$

$R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$

We now proceed through 16 iterations, for $1 \leq i \leq 16$, using function f which operates on two blocks — a data block of 32 bits and a key k_i of 48 bits — to produce a block of 32 bits. Then for i going from 1 to 16 we calculate:

$$L_i = R_{i-1}, R_i = L_{i-1} \otimes f(R_{i-1}, k_i),$$

For $i = 1$, we have

$$k_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$$

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

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

We calculate $E(R_0)$ from R_0 as follows:

$$R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$

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

Note that each block of 4 original bits has been expanded to a block of 6 output bits.

Next in the f calculation, we XOR the output $E(R_{i-1})$ with the key k_i :

$$k_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$$

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

$$k_1 \otimes E(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111$$

Write the previous result, which is 48 bits in the form $k_1 \otimes E(R_{i-1}) = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$, where each B_i is a group of six bits.

$$k_1 \otimes E(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111$$

We now calculate

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

where $S_i(B_i)$ refers to the output of the i -th S-box.

$$\begin{aligned} S_1(B_1)\ S_2(B_2)\ S_3(B_3)\ S_4(B_4)\ S_5(B_5)\ S_6(B_6)\ S_7(B_7)\ S_8(B_8) &= 0101\ 1100\ 1000\ 0010 \\ 1011\ 0101\ 1001\ 0111 \end{aligned}$$

The final stage in the calculation of f is to do permutation P , we get

$$f = 0010\ 0011\ 0100\ 1010\ 1010\ 1001\ 1011\ 1011$$

$$S_0 R_1 = L_0 \otimes f(R_0, k_1)$$

$$= 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111 \otimes 0010\ 0011\ 0100\ 1010\ 1010\ 1001\ 1011\ 1011$$

$$= 1110\ 1111\ 0100\ 1010\ 0110\ 0101\ 0100\ 0100$$

After the first round we get

$$L_i = 1111000010101010111000010101010$$

$$R_i = 11101111010010100110010101000100$$

Now we simply repeat this simple process 15 more times.

$$\begin{aligned}
 L_2 &= 11101111010010100110010101000100 \\
 L_3 &= 11001100000000010111011100001001 \\
 L_4 &= 101000100101110000010111110100 \\
 L_5 &= 01110111001000100000000001000101 \\
 L_6 &= 10001010010011111010011000110111 \\
 L_7 &= 11101001011001111100110101101001 \\
 L_8 &= ?00001?00100?0101?11?0?0000?000? \\
 L_9 &= 11010101011010010100101110010000 \\
 L_{10} &= 00100100011111001100011001111010 \\
 L_{11} &= 10110111110101011101011110110010 \\
 L_{12} &= 11000101011110000011110001111000 \\
 L_{13} &= 01110101101111010001100001011000 \\
 L_{14} &= 00011000110000110001010101011010 \\
 L_{15} &= 11000010100011001001011000001101 \\
 L_{16} &= 01000011010000100011001000110100
 \end{aligned}$$

$$\begin{aligned}
 R_2 &= 11001100000000010111011100001001 \\
 R_3 &= 101000100101110000010111110100 \\
 R_4 &= 01110111001000100000000001000101 \\
 R_5 &= 10001010010011111010011000110111 \\
 R_6 &= 11101001011001111100110101101001 \\
 R_7 &= ?00001?00100?0101?1110?0000?000? \\
 R_8 &= 11010101011010010100101110010000 \\
 R_9 &= 00100100011111001100011001111010 \\
 R_{10} &= 10110111110101011101011110110010 \\
 R_{11} &= 11000101011110000011110001111000 \\
 R_{12} &= 01110101101111010001100001011000 \\
 R_{13} &= 00011000110000110001010101011010 \\
 R_{14} &= 11000010100011001001011000001101 \\
 R_{15} &= 01000011010000100011001000110100 \\
 R_{16} &= 00001010010011001101100110010101
 \end{aligned}$$

We then reverse the order of the two blocks $R_{16}L_{16}$ and apply IP^{-1} with the following result:

$$c = 10000101\ 11101000\ 00010011\ 01010100\ 00001111\ 00001010\ 10110100\ 00000101$$

which in hexadecimal format is 85E813540F0AB405.

Therefore, the encrypted form of $m = 0123456789ABCDEF$ is $c = 85E813540F0AB405$.

References

- Pinzon, Yoan. “Introducción a la criptografía y a la seguridad de la información”, 2013.
- Menezes A., Handbook of applied Cryptografy, 5th Edition. CRC Press, 2001.
- A Graduate Course in Applied Cryptography by D. Boneh and V. Shoup
- H. Delfs and H. Knebl, Introduction to Cryptography, 3rd ed. 2015.
- J. A. Buchmann, Introduction to Cryptography. 2004.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Ch. 31 in Introduction to Algorithms, 3rd ed. 2009.
- B. Lomas de Zamora: Gradi. Hacking desde cero, Fox Andina, 2011.
- Secure Coding Working Group, Japan Smartphone Security Association (JSSEC), Android Application Secure Design/Secure Coding Guidebook, 2017.