# Teoria da Informação

# Trabalho Prático nº 3

# Encriptação

# Data Encryption Standard

# Introdução

**Período de execução**: 3 aulas práticas laboratoriais

**Formato de Entrega**:
**Meta**: entrega final (código completo – não é necessário relatório)

**Prazo de Entrega**:
**Meta: 15 de Dezembro, 23h59**

**Esforço extra aulas previsto**: 15h/aluno

Objectivo: Pretende-se que o aluno adquira sensibilidade para as questões fundamentais relacionadas com os algoritmos de encriptação de chave simétrica.

# Trabalho Prático

O presente trabalho consiste na implementação do algoritmo de encriptação **DES** (Data Encryption Standard). O processo descrito é ilustrado na Figura 1.

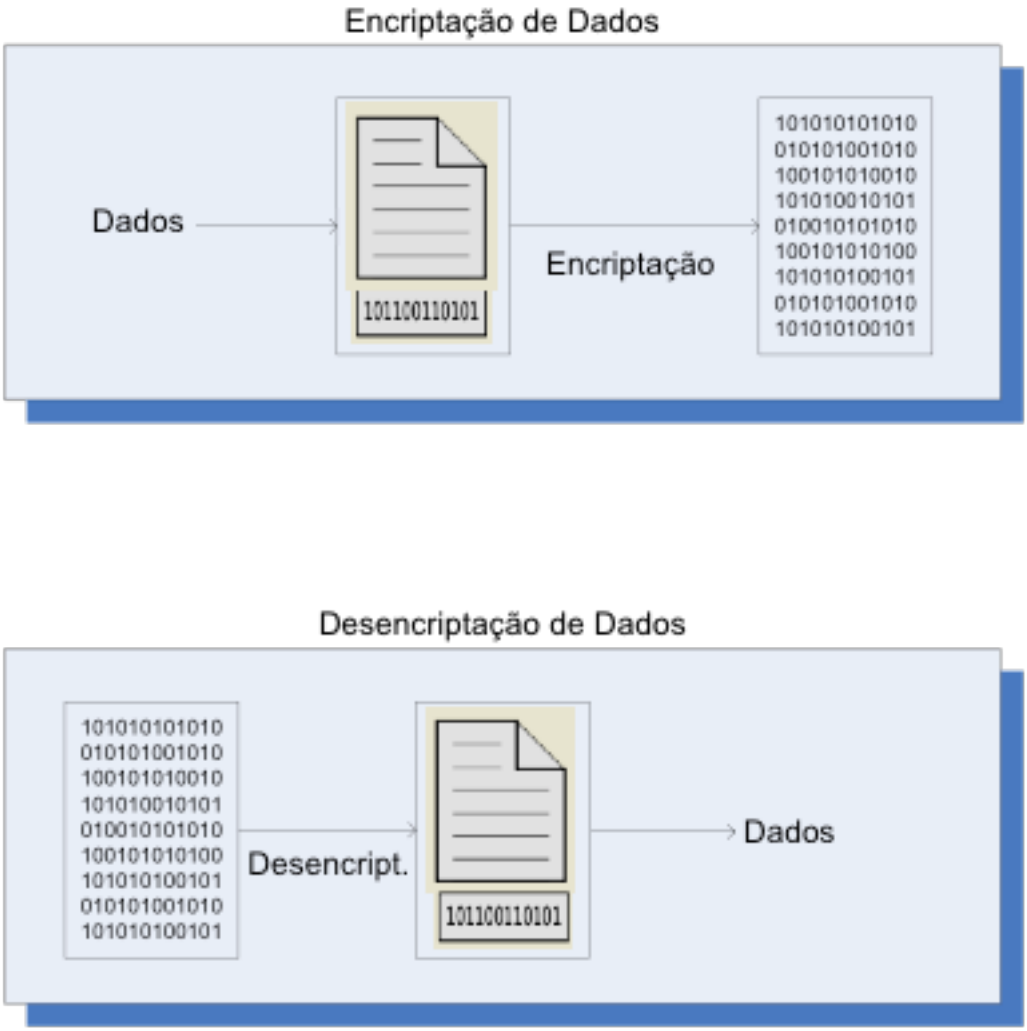

Figura 1. (cima) Encriptação de dados. (baixo) Desencriptação de dados.

## A. Preparação

1. Estudo da teoria sobre encriptação em geral e dos algoritmos DES (ver ficheiro **DES.pdf**, em anexo, e descrição abaixo – secção C).
2. Estudo do código fonte fornecido, como base para trabalho. Deverá estudar as funcionalidades implementadas. É objectivo do trabalho que estenda este código de forma a incluir as restantes etapas necessárias à implementação dos algoritmos.

- Ficheiro **main.c**: ficheiro principal para codificação e descodificação de um ficheiro com base no esquema Hamming (*n*, *k*). Contém um exemplo de utlização do API.
  - Funções do API:
    - `int DES (char* inFileName, unsigned long long key);`
      - Encripta o ficheiro com nome *inFileName* utilizando a chave *key*; devolve código de erro ou 0 na ausência de erro.
    - `int unDES (char* inFileName, unsigned long long key);`
      - Desencripta o ficheiro com nome *inFileName*; devolve código de erro ou 0 na ausência de erro.

- Ficheiro **DES.c**: ficheiro principal com as funções para encriptação/desencriptação e criação/verificação da assinatura. Merecem particular atenção as funções seguintes:
  - `unsigned char* encryptDES(unsigned char* inByteArray, long dim, unsigned long long key, int type)`
    - Encripta (*type* = 0) ou desencripta (*type* = 1) os dados no array *inByteArray*, de dimensão *dim*, com base na chave *key* (nota: o array corresponde, em geral, a dados previamente lidos de um ficheiro)
  - `unsigned long long encryptDESplain(unsigned long long plain, unsigned long long* subKeys)`
    - Encripta a mensagem *plain* (de 64 bits), devolvendo a mensagem cifrada (também com 64 bits), com base em 16 *subKeys* de 48 bits cada (ver função seguinte)
  - `void DESKeySchedule(unsigned long long key, unsigned long long* subKeys)`
    - Gera 16 *subKeys* de 48 bits cada com base na chave *key*

## B. Trabalho a realizar:

Depois de estudado o código fonte, deverá desenvolver as seguintes funções, de acordo com a descrição acima:

- **Encriptação**:
  - `void DESKeySchedule(unsigned long long key, unsigned long long* subKeys)`
    - Gera 16 *subKeys* de 48 bits cada com base na chave *key*
    - Esta função é chamada a partir da função *encryptDES*
  - `unsigned long long encryptDESplain(unsigned long long plain, unsigned long long* subKeys)`
    - Encripta a mensagem *plain* (de 64 bits), devolvendo a mensagem cifrada (também com 64 bits), com base em 16 *subKeys* de 48 bits cada (ver função seguinte)
    - Esta função é chamada a partir da função *encodeData*

**- Desencriptação**:

  - `unsigned char* encryptDES(unsigned char* inByteArray, long dim, unsigned long long key, int type)`
    - Adicionar código a esta função para inverter as subKeys geradas, caso *type* = 1 (i.e., desencriptação)
    - Esta função é chamada a partir da função *DESgeneral*

- **Testes:**
  - Teste o seu código com vários ficheiros e chaves
  - Resultados esperados para a chave 0x0123456789ABCDEF → ver ficheiro **resultados.xls** (resultados de encriptação do ficheiro de teste)
  - Teste de desencriptação: corra o desencriptador sobre o ficheiro FAQ.txt.gz.DES (versão encriptada do ficheiro FAQ.txt.gz) e verifique se o resultado obtido é igual ao original.

# C. Descrição do DES:

## C.1. Algoritmo de Encriptação

O DES é um algoritmo de cifra Feistel que processa de forma sequencial blocos de 64 bits de dados, produzindo para cada bloco tratado 64 bits de dados encriptados. O algoritmo de encriptação baseia-se numa chave secreta de 56 bits e numa sequencia de 16 estágios de transformação semelhantes (vide figura 1).

Usando a chave secreta *K* são geradas 16 chaves secretas $K_i$, de 48 bits cada, que intervêm uma por estágio de transformação. Cada bloco de 64 bits é inicialmente permutado de acordo com a tabela 1, isto é, seja $x = x_1 x_2 x_3 \ldots x_{64}$ a palavra de 64 bits de entrada, o resultado deste estágio será $x' = x_{IP(1)} x_{IP(2)} x_{IP(3)} \ldots x_{IP(64)} = x_{58} x_{50} x_{42} \ldots x_7$. Posteriormente, a palavra transformada é dividida em parte esquerda $L_0$ (32 bits mais significativos) e parte direita $R_0$ (32 bits menos significativos), iniciando-se os estágios de transformação destas palavras.

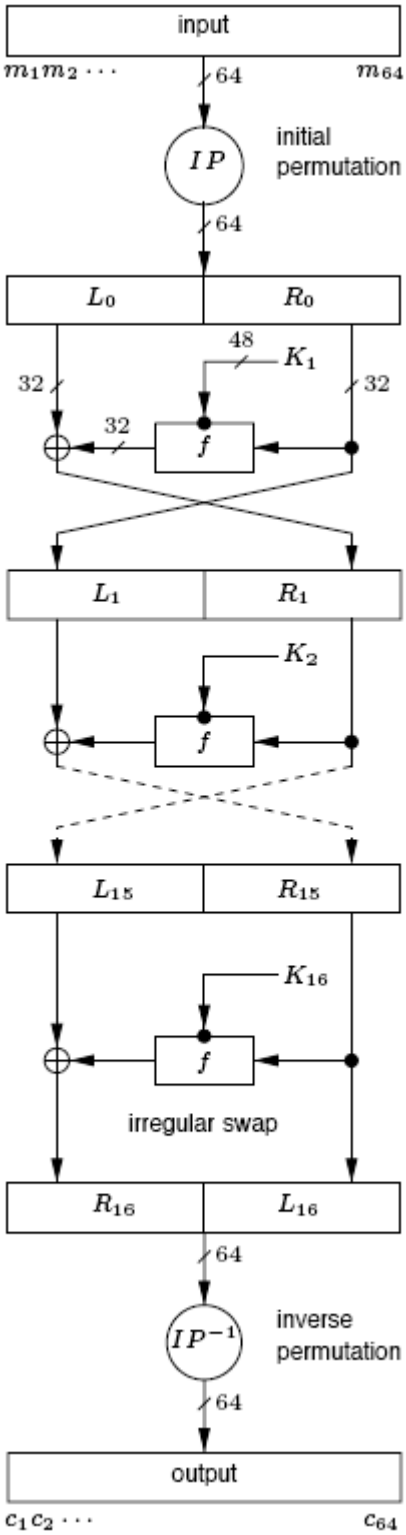


Figura 1: Estrutura do Algoritmo DES.

| 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 |

| $IP^{-1}$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

Tabela 1: Definição das permutações iniciais e finais do DES.

Cada estágio de transformação no DES é funcionalmente equivalente, transformando os resultados $L_{i-1}$ e $R_{i-1}$ do estágio anterior de acordo com as equações seguintes:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f\left(R_{i-1}, K_i\right)$$

Nesta equação a função *f* é composta por blocos de transformação (vide figura 2) de expansão da palavra $R_{i-1}$ de 32 bits para 48 bits, seguida de uma substituição fixa por recurso às designadas S-boxes, finalizando com uma permutação fixa. Isto é,

$$f\left(R_{i-1}, K_i\right) = P\left(S\left(E\left(R_{i-1}\right) \oplus K_i\right)\right)$$

em que as funções E, P e S estão definidas nas tabelas 2 e 3, respectivamente. Na função S, os índices da coluna e linha são determinados para cada bloco de 6 bits, $b_1 b_2 b_3 b_4 b_5 b_6$ , de acordo com a fórmula:

$$c = b_2 b_3 b_4 b_5$$
$$l = 2b_1 + b_6$$

| *E* | | | | | |
|---|---|---|---|---|---|
| 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 |

| *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 |

Tabela 2: Definição das funções E e P.

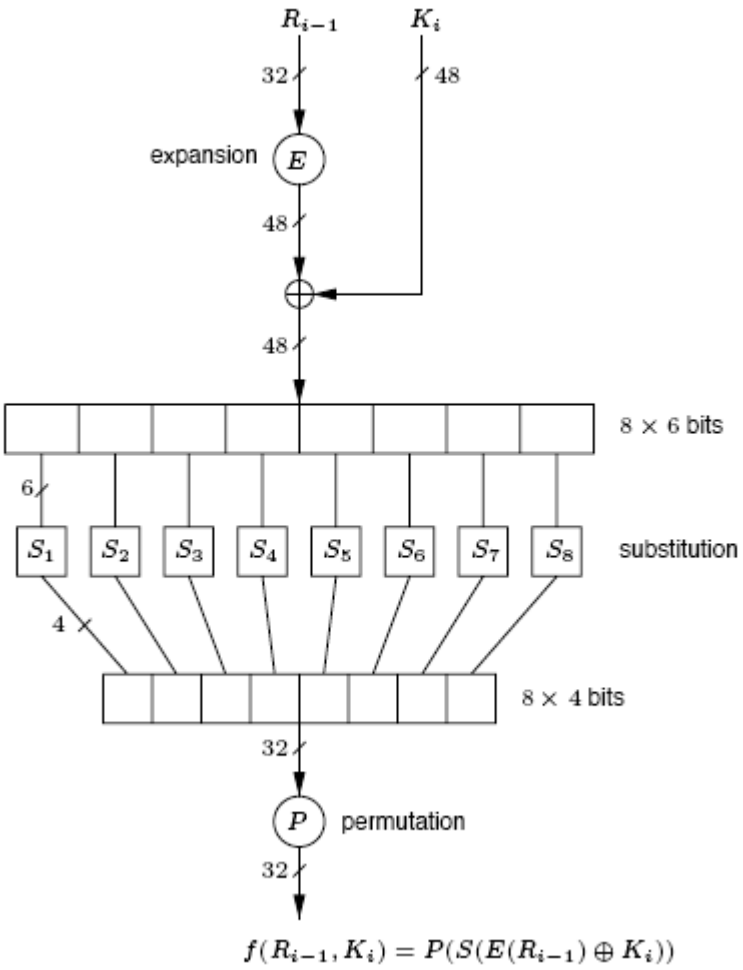


Figura 2: Estrutura interna da função de transformação do DES.

| row | column number | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| $S_1$ | | | | | | | | | | | | | | | | |
| [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 |
| $S_2$ | | | | | | | | | | | | | | | | |
| [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 |
| $S_3$ | | | | | | | | | | | | | | | | |
| [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 |
| $S_4$ | | | | | | | | | | | | | | | | |
| [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] | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |
| $S_5$ | | | | | | | | | | | | | | | | |
| [0] | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| [1] | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| [2] | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| [3] | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |
| $S_6$ | | | | | | | | | | | | | | | | |
| [0] | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| [1] | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| [2] | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| [3] | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |
| $S_7$ | | | | | | | | | | | | | | | | |
| [0] | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| [1] | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| [2] | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| [3] | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |
| $S_8$ | | | | | | | | | | | | | | | | |
| [0] | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| [1] | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| [2] | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| [3] | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

Tabela 3: As S-Boxes do DES.

## C.2. Geração de Sub-chaves

Embora a chave secreta do DES seja de 56 bits, esta é normalmente apresentada com 64 bits em que cada 8º bit representa um bit (8º, 16º, 24º, ..., 64º) de paridade. Assim, dada uma chave de 64 bits nas condições acima enunciadas, são geradas 16 sub-chaves, uma por estágio de transformação do DES. A geração de cada chave envolve sempre duas operações de rotação e uma operação de permutação e selecção. A estrutura geral do algoritmo pode ser visualizada na figura seguinte:

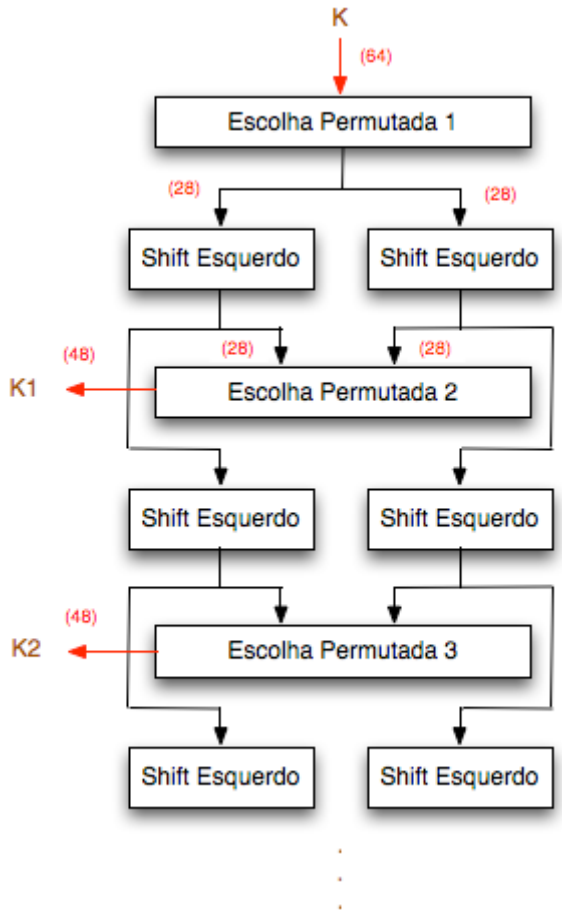


Figura 3: Geração das Sub-chaves no DES.

Os passos do algoritmo são os descritos no fluxo grama seguinte:

Entrada: uma chave de 64 bits, $K = k_1 k_2 k_3 .... k_{64}$
Saída: dezasseis chaves de 48 bits cada, isto é, $K_i = k_1 k_2 k_3 .... k_{48}, 1 \le i \le 16$

1. Definir os shifts circulares $v_i, 1 \le i \le 16$ tal que

$$v_i = \begin{cases} 1 \Leftarrow i \in \{1,2,9,16\} \\ 2 \Leftarrow \text{ caso contrário} \end{cases}$$

2. Permutar e dividir a chave em duas sequências de 28 bits, C0 e D0 de acordo com a função de permutação PC1 da tabela 4.
3. Geração das 16 Sub-Chaves: Para *i* de 1 até 16, determinar a chave Ki fazendo:
    i. Rodar circularmente $C_i$, isto é, $C_i \leftarrow C_{i-1}\ rot\ v_i$
    ii. Rodar circularmente $D_i$, isto é, $D_i \leftarrow D_{i-1}\ rot\ v_i$
    iii. Compor a chave usando a função de escolha e permutação PC2 (vide tabela 5), isto é, $K_i = PC2(C_i, D_i)$. Seja $b_1 b_2 b_3 ....b_{56} = C_i \| D_i$, então, de acordo com a tabela 5, $K_i = b_{14} b_{17} ....b_{32}$.

| PC1 | | | | | | |
|---|---|---|---|---|---|---|
| 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 |
| above for $C_i$; below for $D_i$ | | | | | | |
| 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 | 12 | 4 |

Tabela 4: Definição da função de permutação PC1.

| PC2 | | | | | |
|---|---|---|---|---|---|
| 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 |

Tabela 5: Definição da função de permutação PC2.

**C.1. Algoritmo de Desencriptação**

O algoritmo de desencriptação é semelhante ao de encriptação, obrigando a que as chaves intervenham por ordem inversa, isto é, $K_{16}, K_{15}, ..., K_1$. O efeito de $IP^{-1}$ é cancelado pelo módulo *IP*. Adicionalmente, observa-se que

$$\begin{aligned} L_{i-1} &= R_i \oplus F\left(R_{i-1}, K_i\right) \\ &= R_i \oplus F\left(L_i, K_i\right) \end{aligned}$$

donde

$$R_i = L_{i-1} \oplus F\left(L_i, K_i\right)$$