



**Universidade de Brasília**  
Departamento de Ciência da Computação  
Segurança Computacional

## **Relatório:**

Trabalho Prático 1  
S-DES

Matheus Lopes de Souza  
Aquila Macedo

Mat:19/0043831  
Mat:20/2021800

Professor(a):  
Lorena Borges

**20 de Dezembro de 2024**

# 1 S-DES (Simplified Data Encryption Standard)

É uma versão simplificada do algoritmo DES (Data Encryption Standard). Possui características semelhantes ao DES, porém com menos parâmetros. Esse algoritmo foi criado para fins educacionais, tornando o entendimento ao DES mais simples. A encriptação do S-DES utiliza um bloco de dados de 8 bits e uma chave de 10 bits como entrada e gera um dado cifrado de 8 bits como saída. A decifração utiliza os 8 bits cifrados e a mesma chave para obter o bloco de dados original. Para que os passos anteriores sejam corretamente executados, o S-DES precisa realizar a geração de sub-chaves através da chave de 10 bits da entrada.

## 2 Implementação

Utilizamos a linguagem C++ para o desenvolvimento do algoritmo S-DES, tanto para geração de chaves quanto para encriptar/deciprar. Os tópicos a seguir demonstrarão como foi criada cada parte do algoritmo.

### 2.1 Geração de chaves

A seguir demonstraremos passo a passo das funções que compõem a geração das chaves.

1. **Função P10:** Realiza a permutação da chave de 10 bits baseado na tabela:

```
1 for (int i = 0; i < 10; i++) {  
2     permuted_key[i] = key_10bit[P10[i] - 1];  
3 }
```

2. **Deslocamento circular (K1):** Realiza a divisão da chave em duas metades e aplica, em cada, o deslocamento circular de 1 bit para a esquerda utilizando a função:

```
1 shifted_left = circular_shift(left, 1)  
2 shifted_right = circular_shift(right, 1);
```

3. **Função P8 (K1):** Com os bits deslocados, é realizada a combinação entre as duas metades para que seja aplicada a função de permutação para 8 bits:

```
1 for (int i = 0; i < 8; i++) {  
2     subkey_1[i] = first_combined_key[P8[i] - 1];  
3 }
```

4. **Deslocamento circular (K2):** Realiza o mesmo procedimento que gerou a chave K1, porém o deslocamento circular é de 2 bits para esquerda:

```
1 shifted_left = circular_shift(left, 2)  
2 shifted_right = circular_shift(right, 2);
```

5. **Função P8 (K2):** É feito novamente a permutação de 8 bits, porém utilizando as metades que foram deslocadas de 2 bits:

```
1 for (int i = 0; i < 8; i++) {  
2     subkey_2[i] = second_combined_key[P8[i] - 1];  
3 }
```

## 2.2 Encriptação/Decriptação

O algoritmo possui como entrada uma chave de 10 bits, essa chave é manipulada para produzir 2 subchaves de 8 bits que serão utilizadas para encriptar/decriptar.

### 2.2.1 Permutação Inicial

Este passo realiza a permutação do bloco de dados em claro (encriptação) ou cifrado (decriptação) baseado na tabela IP:

```
1 for (int i = 0; i < 8; i++) {  
2     permuted_bits[i] = plaintext[IP[i] - 1];  
3 }
```

### 2.2.2 Divisão em metades

É feita a divisão do bloco de entrada em duas metades:

```
1 left_half(input_bits.begin(), input_bits.begin() + 4);  
2 right_half(input_bits.begin() + 4, input_bits.end());  
3 }
```

### 2.2.3 Rodadas de Feistel

Nessa etapa são executadas duas rodadas de Feistel, que possui os seguintes passos em cada rodada:

1. **Função  $F_k$ :** Uma função não-linear recebe a metade da direita (R) combinada com a subchave da rodada (K1 na primeira rodada e K2 na segunda rodada). Expansão/permutação, S-Boxes (substituições) e uma permutação P4 são executadas nesse momento;
2. **XOR:** A saída da função  $F_k$  é combinada com a metade da esquerda (L) utilizando a operação XOR bit a bit;
3. **Troca:** As metades (L) e (R) são trocadas após o final da primeira rodada.

### 2.2.4 Permutação Inicial Inversa

Após as Rodadas de Feistel, é aplicada a permutação inicial inversa para obter o bloco cifrado (encriptação) ou em claro (decriptação):

```
1 for (int i = 0; i < 8; i++) {  
2     ciphertext[i] = second_round_output[IP_INVERSE[i] - 1];  
3 }
```

## 3 Cenários de Teste

### 3.1 Execução

:

- Compilar o arquivo:

```
1 make
```

ou

```
1 g++ main.cpp sdes.cpp sdes_demo.cpp -o sdes
```

- Para executar:

```
1 ./sdes
```

### 3.2 Parâmetros e Dados

Vamos testar o algoritmo S-DES implementado utilizando o seguinte contrato de entrada e saída:

#### 1. Entrada:

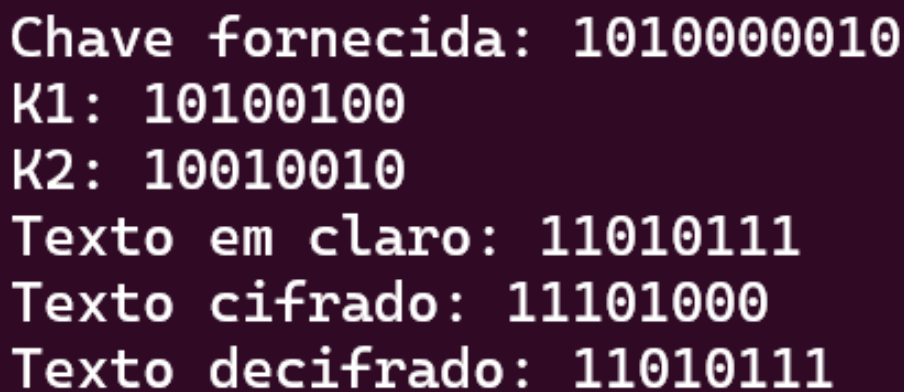
- Chave de 10 bits: 1010000010
- Bloco de dados de 8 bits: 11010111

#### 2. Saída esperada:

- Bloco encriptado.: 1010000010
- Bloco descriptado: 11010111

### 3.3 Resultado

A imagem abaixo mostra o resultado da execução do algoritmo utilizando os dados de entrada propostos:



```
Chave fornecida: 1010000010
K1: 10100100
K2: 10010010
Texto em claro: 11010111
Texto cifrado: 11101000
Texto decifrado: 11010111
```