Universidade de Brasília

Departamento de Ciência da Computação

Projeto 1: Implementação do algoritmo S-DES

Autor:

Rafael Dias Ghiorzi - 232006144

Disciplina de Segurança Computacional

Professora Lorena Borges

Brasília
13 de maio de 2025

1 Introdução

A criptografia simétrica é um dos pilares da segurança da informação moderna. Ela é caracterizada pelo uso de uma mesma chave tanto para cifragem quanto para decifragem de dados. Nesse contexto, o SDES (*Simplifiend Data Encryption Standard*) surge como uma versão didática e simplificada do algoritmo DES (*Data Encryption Standard*), que foi amplamente utilizado nas décadas de 1970 a 1990 como padrão de criptografia, mas que atualmente foi substituido pelo AES (*Advanced Encryption Standard*), dada a necessidade de ferramentas mais robustas de criptografia.

O SDES foi desenvolvido com propósito educacional, visando facilitar o entendimento dos conceitos fundamentais da criptografia de bloco sem a complexidade completo do DES original. Sua implementação permite compreender os princípios da substituição, permutação e confusão nos sistemas criptográficos.

2 Fundamentação Teórica

A criptografia de bloco funciona transformando dados de entrada em blocos de tamanho fixo a partir de uma série de operações determinadas por uma chave secreta. O SDES, assim como o DES, é classificado como uma cifra de bloco.

O SDES difere do DES completo da seguinte forma:

Característica	SDES	DES
Tamanho do bloco	8 bits	64 bits
Tamanho da chave	10 bits	56 bits efetivos (64 bits iniciais)
Número de rodadas	2	16
S-Boxes	2 de 2x4	8 de 6x4
Expansão	De 4 para 8 bits	De 32 para 48 bits

Tabela 1: Diferenças entre SDES e DES

Os principais componentes do SDES são:

- **Permutações:** Realizam um rearranjo arbitrário dos bits de entrada. No SDES, temos a permutação inicial IP e sua inversa IP^{-1}, P10, P8 e P4.
- **Caixas de substituição:** Substituição de padrões de bits por outros valores predefinidos. Temos duas caixas, S0 e S1 no algoritmo, cada uma mapeando 4 bits para 2 bits de saída.

- **Função de Feistel:** Estrtutura que divide o bloco em duas metades e aplica uma função específica (função F) em uma das metades e depois combina o resultado com outra metade a partir da operação XOR.

3 Visão geral do algoritmo

3.1 Estrutura

O algoritmo segue a seguinte estrutura:

1. Geração de Subchaves
2. Permutação inicial IP do bloco de bits de entrada
3. Aplicação da primeira rodada de Feistel com a subchave K_1
4. Troca das metades (switch)
5. Aplicação da segunda rodada de Feistel com a subchave K_2
6. Permutação final inversa IP^{-1}

3.2 Geração das Subchaves

O processo de geração de K_1 e K_2 a partiri da chave original segue os passos a seguit:

1. Aplicação da permutação P10
2. Divisão em dois blocos de 5 bits E0 e D0
3. Aplicando um deslocamento circular à esquerda nas duas metades
4. Aplicar a permutação P8 no resultado para obter K_1
5. Aplicar dois deslocamentos circulares nas duas metades
6. Aplicar novamente a permutação P8 para obter K_2

3.3 Processo de Cifragem

1. Aplicação da IP
2. Divisão do bloco em dois, E0 e D0, de 4 bits
3. Aplicação da função F em D0 com a subchave K_1
4. Combinar o resultado com E0 a partir de um XOR para obter D1
5. O novo bloco será formado por D0 (agora E1) e D1
6. Aplicação da função F ao novo bloco com a subchave K_2)
7. Combinação do resultado com E1 a partir de um XOR para obter D2
8. Bloco final (D2, E1)
9. Aplicação da permutação final inversa IP^{-1}

A função F, é composta por:

1. Expansão do bloco de 4 bits com a P8
2. XOR com a subchave da rodada
3. Divisão em duas partes para aplicação das S-Boxes S0 e S1
4. Permutação P4 no resultado

3.4 Implementação de ECB e CBC

Existem dois modos de operação que podem ser implementados em cifras de blocos:

- **Eletronic Codebook (ECB):** Cada bloco é cifrado independentemente. Essa forma de implementação não é considerada segura por manter os padrões de repetição caso alguns blocos sejam semelhantes, o que favorece a interpretação e quebra da cifra.
- **Cipher Block Chaining (CBC):** Cada bloco é combinado com o bloco de texto cifrado anterior a partir de um XOR. Para o primeiro bloco, é utilizado um vetor de inicialização. Esse método garante a perda das repetições e é muito mais seguro do que as implementações simples vistas.

De fato, o modo CBC proporciona maior segurança ao remover padrões de repetição, pois blocos idênticos de texto claro resultarão em blocos diferentes de texto cifrado, devido à dependência do bloco anterior.

4 Implementação

A implementação do algoritmo SDES foi realizada em Python, utilizando o ambiente de um jupyter notebook para maior controle das etapas. O Python foi escolhido por sua simplicidade de sintaxe e fácil entendimento do código.

A implementação foi organizada em componentes modulares para facilitar a compreensão e manutenção. O código completo pode ser encontrado no seguinte repositório do GitHub: Implementação do SDES

Segue alguns trechos relevantes do código para primeiras impressões:

Obs: note que as declarações de variáveis e funções não estão presentes. Isso foi feito para manter o código limpo e sucinto. No link disponibilizado acima, é possível encontrar os códigos completos

```
# Função de geração das chaves
def key_gen(key: list) -> tuple:
    # initial permutation
    key = permutation(key, P10)
    # creating left and right side
    left, right = key[:5], key[5:]
    # single circular left shift
    left, right = shift(left, 1), shift(right, 1)
    # first subkey K1
    k1 = permutation(left+right, P8)
    # double circular left shift
    left, right = shift(left, 2), shift(right, 2)
    # second subkey k2
    k2 = permutation(left+right, P8)

    return (k1, k2)

# F mapping function

def F_map(key: list, bits: list) -> list:
    # expanding the bits
    bits = permutation(bits, EXPAND)
    bits = xor(key, bits)

    # selecting and joining sbox bits
    sbox1 = sbox(bits[:4], S0)
```

```
    sbox2 = sbox(bits[4:], S1)

    # return the permutation of the sboxes
    bits = permutation(sbox1 + sbox2, P4)

    return bits

# Feistel encryption function

def Fk(bits: list, key: list) -> list:
    left, right = bits[:4], bits[4:]
    right_F = F_map(key, right)  # F map function
    bits = xor(left, right_F) + right

    return bits

# main SDES implementation
def sdes(bits: list, key: list) -> str:

    # generating the subkeys
    k1, k2 = key_gen(key)
    # initial permutation
    bits = permutation(bits, IP)
    # first Feistel round
    bits = Fk(bits, k1)
    # Swap left and right halves
    bits = bits[4:] + bits[:4]
    # second Feistel round
    bits = Fk(bits, k2)
    # final permutation
    bits = permutation(bits, IP_1)

    # final result
    return bits
```

5 Análise dos Resultados

A implementação do algoritmo SDES foi testada com a entrada pedida pela professora. A chave é a mesma para ambos os métodos: 1010000010. Os resultados obtidos mostram o comportamento esperado de um algoritmo de criptografia simétrica:

1. **Cifragem simples**
    - Texto claro: 11010111
    - Texto cifrado: 10101000
2. **Cifragem no modo ECB**
    - Texto claro: 11010111011011001011101011110000
    - Texto cifrado: 10101000000011010010111001101101
3. **Cifragem no modo ECB**
    - Texto claro: 11010111011011001011101011110000
    - Vetor de inicialização: 01010101
    - Texto cifrado: 00001011101010011001101101101010

6 Conclusão

Após a implementação e teste do SDES, é possível, a partir da simplicidade didática, compreender os princípios básicos da criptografia de bloco, estrutura de Feistel e os elementos essenciais dos algoritmos (permutação, substituição e confusão).

Além disso, pode-se observar que o SDES realmente é um algoritmo puramente educativo, dado que seu tamanho reduzido e sua pequena complexidade não são boas para aplicação no mundo real. O espaço amostral de $2^{10} = 1024$ (chave de 10 bits) é muito pequeno e pode ser facilmente quebrado com força bruta.

Apesar disso, esse trabalho apresentou uma implementação completa do algortimo, abordando desde os fundamentos teóricos até a implementação prática. A implementação manteve o foco na clareza e didática, priorizando a compreensão dos conceitos fundamentais.