Relatório do Trabalho de Segurança

20 de dezembro de 2024

Aluno: Guilherme Araújo de Oliveira / 190125748

Aluno: Guilherme Praxedes Franco / 180136780

Introdução:

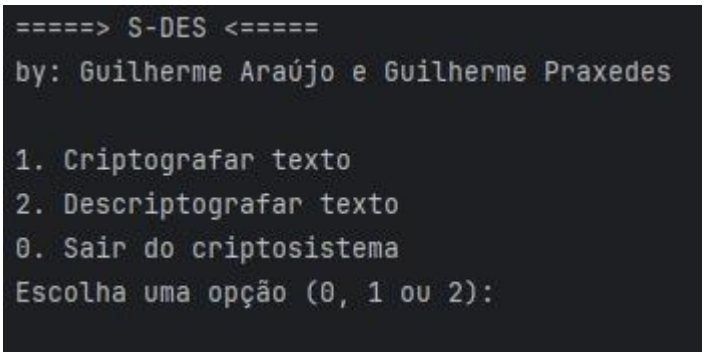
No trabalho iremos desenvolver um sistema de encriptação e desencriptação utilizando o algoritmo Simplified Data Encryption Standard (S-DES), versão simplificada do DES, usada para fins educacionais pelo fato de possuir uma estrutura acessível.

Foram implementados passos como geração de chaves, permutações e substituições (S-Boxes), essenciais para o funcionamento do S-DES. Tudo isso foi feito através da utilização do Python, linguagem escolhida pela dupla para a confecção do código. O sistema permite aplicar conceitos teóricos e entender os mecanismos de segurança.

Assim, o trabalho reforça conhecimentos sobre criptografia e demonstra a aplicabilidade dessas técnicas no desenvolvimento de soluções seguras em um ambiente digital desafiador.

Desenvolvimento:

Para que seja dado início, foi feito um pequeno prompt para que o usuário possa escolher qual o tipo de ação será realizado, então temos como prompt inicial:



O código então receberá o dado cedido pelo usuário e fará a ação solicitada. Foi feito um esquema do qual, através de arquivos Json, sejam salvas as chaves e subchaves no decorrer do código.

```python
def main() -> None:  1 usage  new *
    print("\t=====> S-DES <=====\nby: Guilherme Araújo e Guilherme Praxedes")
    while True:
        print("\n1. Criptografar texto\n2. Descriptografar texto\n0. Sair do criptosistema")
        choice = input("Escolha uma opção (0, 1 ou 2): ")

        if choice == "1":
            encrypt()

        elif choice == "2":
            decrypt()

        elif choice == "0":
            break
        else:
            print("Opção inválida. Escolha entre 0, 1 e 2.")

if __name__ == "__main__":
    main()
```

O código para a entrada de dados.

```python
def encrypt():  1 usage  guilhermea23 *
    plaintext = input("Digite o texto para criptografar: ")
    binary_text = to_bin(plaintext)

    key = gen_key_10b()
    k1, k2 = generate_subkeys(key)

    encrypted_blocks = [encrypt_block(block, k1, k2) for block in binary_text]

    with open("key.json", "w") as file:
        json.dump(key, file)

    with open("encrypted_blocks.json", "w") as file:
        json.dump(encrypted_blocks, file)

    print("\nTexto criptografado salvo em 'encrypted_blocks.json'.")
    return encrypted_blocks
```

Código para a criptografia do texto.

Para que seja possível criptografar e descriptografar, seguiremos a ideia de criar blocos de chaves e dessa forma realizar a ação.

```python
def to_bin(text):  1 usage  guilhermea23
    return [format(ord(char), '08b') for char in text]

def to_text(binary_list):  1 usage  guilhermea23
    return ''.join(chr(int(b, 2)) for b in binary_list)
```

Quando o texto é convertido para binário o código já realiza o ajuste necessário para que o bloco seja de 8bits.

Para a continuação da execução do código é necessário que seja entendido que, para que o S-DES realize as suas ações, são necessárias as presenças de permutações durante o código e a geração de uma chave de 10bits, para que sejam geradas duas subchaves de 8bits.

Após isso, é realizado uma permutação nos blocos de acordo com a permutação anteriormente gerada, depois disso, são geradas as subchaves de 8bits.

```
def gen_key_10b():  1 usage  guilhermea23
    return ''.join(random.choice('01') for _ in range(10))

def generate_permutation(size):  guilhermea23
    perm = list(range(1, size + 1))
    random.shuffle(perm)
    return perm
```

```
def apply_permutation(data, permutation):  7 usages  guilhermea23
    return ''.join(data[i - 1] for i in permutation)

def generate_subkeys(key):  2 usages  guilhermea23
    p10 = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]
    permuted_key = apply_permutation(key, p10)

    left_half, right_half = permuted_key[:5], permuted_key[5:]
    left_half_ls1 = left_half[1:] + left_half[0]
    right_half_ls1 = right_half[1:] + right_half[0]
    key_ls1 = left_half_ls1 + right_half_ls1

    p8 = [6, 3, 7, 4, 8, 5, 10, 9]
    k1 = apply_permutation(key_ls1, p8)

    left_half_ls2 = left_half_ls1[2:] + left_half_ls1[:2]
    right_half_ls2 = right_half_ls1[2:] + right_half_ls1[:2]
    key_ls2 = left_half_ls2 + right_half_ls2

    k2 = apply_permutation(key_ls2, p8)

    return k1, k2
```

Abaixo está exemplificado como foi gerada algumas permutações através de funções intermediárias.

```
def initial_permutation(data):  1 usage  guilhermea23
    ip = [2, 6, 3, 1, 4, 8, 5, 7]
    return apply_permutation(data, ip)

def inverse_permutation(data):  1 usage  guilhermea23
    ip_inverse = [4, 1, 3, 5, 7, 2, 8, 6]
    return apply_permutation(data, ip_inverse)
```

Durante o código, foram utilizadas muitas funções do Python das quais facilitam a realização do S-DES, para que não fiquem dúvidas, faremos uma breve explicação do que cada uma faz:

to_bin(text): Converte um texto em uma lista de strings binárias de 8 bits para cada caractere.

to_text(binary_list): Converte uma lista de strings binárias de 8 bits de volta para o texto original.

gen_key_10b(): Gera uma chave binária aleatória de 10 bits.

generate_permutation(size): Gera uma permutação aleatória para um dado tamanho.

apply_permutation(data, permutation): Aplica uma permutação fornecida nos dados binários.

generate_subkeys(key): Gera duas subchaves de 8 bits a partir de uma chave de 10 bits usando as permutações e deslocamentos específicos.

initial_permutation(data): Aplica a permutação inicial (IP) nos dados de entrada.

inverse_permutation(data): Aplica a permutação inversa (IP-1) nos dados de entrada.

feistel_function(right_half, subkey): Aplica a função Feistel sobre a metade direita dos dados com uma subchave fornecida.

encrypt_block(block, k1, k2): Criptografa um bloco de dados usando o algoritmo S-DES com duas subchaves (k1 e k2).

encrypt(): Realiza a criptografia de um texto fornecido pelo usuário, gera a chave, subchaves e salva o resultado criptografado em arquivos.

decrypt(): Descriptografa o texto criptografado usando a chave e subchaves carregadas de arquivos, e exibe o texto original.

main(): Controla o fluxo do programa, oferecendo opções de criptografar ou descriptografar texto, e chama as funções apropriadas conforme a escolha do usuário.

sbox_lookup(bits, sbox): Função interna à feistel_function() que realiza a busca nas S-boxes. Recebe os bits de entrada (4 bits) e a S-box correspondente (S0 ou S1), e retorna o valor substituído pela S-box após aplicar as regras de linha e coluna.

Por fim, abaixo fica uma imagem do código que fará a Cifra de Feistel, cifra essa que é bem utilizada pelos algoritmos S-DES, ela divide o texto em duas metades, aplicando uma função de transformação a uma metade e combinando o resultado com a outra metade usando uma operação de XOR.

```
def feistel_function(right_half, subkey):  2 usages  guilhermea23
    ep = [4, 1, 2, 3, 2, 3, 4, 1]
    expanded_half = apply_permutation(right_half, ep)

    xor_result = ''.join(str(int(b1) ^ int(b2)) for b1, b2 in zip(expanded_half, subkey))

    s0 = [
        ['01', '00', '11', '10'],
        ['11', '10', '01', '00'],
        ['00', '10', '01', '11'],
        ['11', '01', '11', '10']
    ]

    s1 = [
        ['00', '01', '10', '11'],
        ['10', '00', '01', '11'],
        ['11', '00', '01', '00'],
        ['10', '01', '00', '11']
    ]

    def sbox_lookup(bits, sbox):  guilhermea23
        row = int(bits[0] + bits[3], 2)
        col = int(bits[1] + bits[2], 2)
        return sbox[row][col]

    left_bits = xor_result[:4]
    right_bits = xor_result[4:]
    s0_result = sbox_lookup(left_bits, s0)
    s1_result = sbox_lookup(right_bits, s1)

    combined = s0_result + s1_result

    p4 = [2, 4, 3, 1]
    return apply_permutation(combined, p4)
```

Conclusão:

Este trabalho desenvolveu um sistema de encriptação e desencriptação usando o algoritmo Simplified Data Encryption Standard (S-DES), uma versão simplificada do DES, voltada para fins educacionais.

Foram implementadas etapas como geração de chaves, permutações e substituições (S-Boxes), fundamentais para o funcionamento do S-DES. O sistema permitiu aplicar conceitos teóricos e entender mecanismos de segurança.

Concluindo, o projeto evidenciou a importância dos princípios de criptografia na proteção de informações digitais e consolidou conhecimentos teóricos e práticos sobre segurança computacional.