

Laboratory work 4:
Cryptography and Security
Block ciphers. The DES algorithm

Elaborated:
st. gr. FAF-213

Afteni Maria

Verified:
asist. univ.

Cătălin Mîțu

Chișinău - 2023

ALGORITHM ANALYSIS

Tasks:

Given the key of the DES algorithm (8 symbols), to determine K^+ .

IMPLEMENTATION

First I declared the standart tables that are needed for the DES algorithm. These tables, like "initial_permutation", "pc_1" and "pc_2" are used in the context of the Data Encryption Standard (DES) algorithm to permute and manipulate data bits for encryption. They define specific bit orders and shuffling patterns to create cryptographic keys and perform data transformations.

Initial permutation (IP) table

```
initial_permutation = [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]
```

Permutation choice 1 (PC1) table

```
pc_1 = [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,
        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]
```

Permutation choice 2 (PC2) table

```
pc_2 = [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]
```

The "shift_schedule" indicates how many bits to shift left in each round of the DES algorithm, crucial for key generation.

Left shift values for each round

```
shift_schedule = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]
```

In the 'generate_key' function, I convert the input key into binary. Next, I apply the PC1 permutation by rearranging the bits according to the given table and store the result in 'pc1_result'. Finally, I pass the 'pc1_result' to the 'split_key' function to generate a set of round keys.

```
def generate_key(key):
    key_binary = ""
    for c in key:
        key_binary += ".join(format(ord(c), '08b'))

    pc1_result = []
    for i in range(56):
        pc1_result.append(key_binary[pc_1[i] - 1])

    return split_key(pc1_result)
```

In 'split_key' function, I'm processing a 56-bit key for encryption. I split it into left and right halves, shift them, and then combine them. After combining, I apply a table permutation (PC2) to generate a 48-bit round key. I repeat this process for multiple rounds, storing each round key in a list, and finally return the list of 48-bit keys for encryption.

```
def split_key(key):
    round_key = 0
    permuted_key = []
    for shift in shift_schedule:
        # Split the 56-bit key into left and right halves
        left_half = key[:28]
        right_half = key[28:]

        # Perform circular left shift on both halves
        left_half = left_half[shift:] + left_half[:shift]
        right_half = right_half[shift:] + right_half[:shift]

        # Combine the halves and apply PC2 to get a 48-bit round key
        round_key = left_half + right_half
        pc2_result = []
        for i in range(48):
            pc2_result.append(round_key[pc_2[i] - 1])

        permuted_key.append(".join(pc2_result))

        # Update the key for the next round
        key = left_half + right_half

    return permuted_key
```

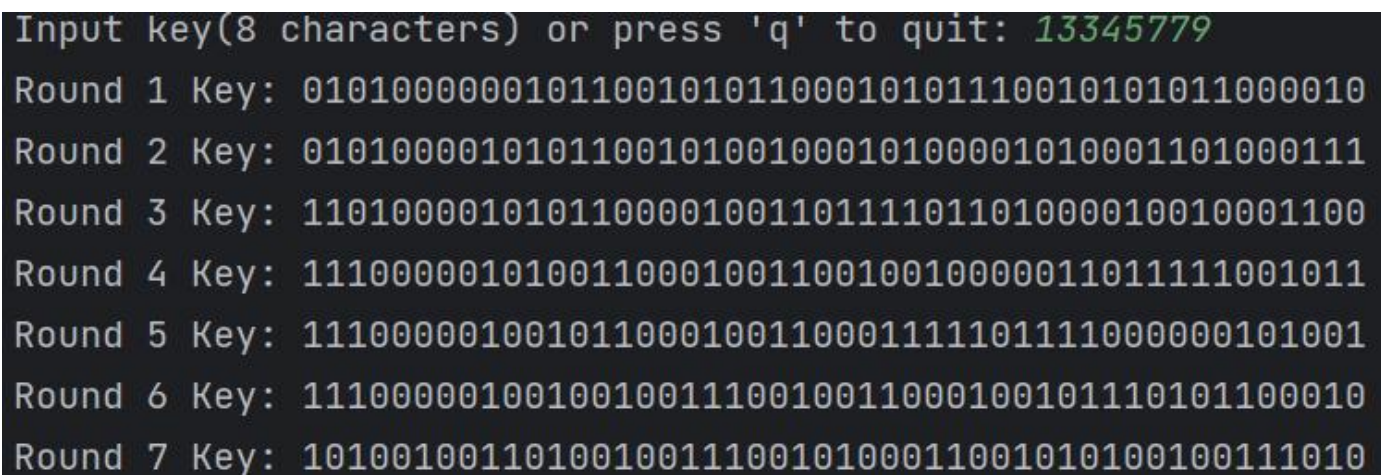
Finally, I've created a program that takes an 8-character input key, generates round keys from it using the 'generate_key' function, and prints them for a DES encryption algorithm. I use a while loop to continuously prompt the user for input, break the loop if 'q' is entered, and handle invalid input by checking the key's length.

```
def main():
    while True:
        key = input("Input key(8 characters) or press 'q' to quit: ")
        if key == "q":
            break
        if len(key) > 8 or len(key) < 8:
            print("Invalid key! Key should contain 8 characters.")
            continue

        permuted_key = generate_key(key)

        for i, round_key in enumerate(permuted_key):
            print(f"Round {i + 1} Key: {round_key}")
```

Example of execution:



```
Input key(8 characters) or press 'q' to quit: 13345779
Round 1 Key: 010100000010110010101100010101110010101011000010
Round 2 Key: 010100001010110010100100010100001010001101000111
Round 3 Key: 110100001010110000100110111101101000010010001100
Round 4 Key: 111000001010011000100110010010000011011111001011
Round 5 Key: 111000001001011000100110001111101111000000101001
Round 6 Key: 111000001001001001110010011000100101110101100010
Round 7 Key: 101001001101001001110010100011001010100100111010
```

CONCLUSIONS

In this laboratory work, I focused on the DES (Data Encryption Standard) algorithm and its key generation process. The primary objective was to determine the K^+ (K plus) value, which represents the derived round keys used in the DES encryption process.

I started by taking an 8-symbol key as input, and then applied various permutation tables and circular shifts as per the DES specification to generate the 16 round keys. These round keys are essential for encrypting and decrypting data using the DES algorithm.

Through this exercise, I gained a better understanding of how DES key generation works and the importance of careful key management in cryptography. This knowledge is fundamental for security applications where data protection is crucial. Overall, the laboratory work provided valuable insights into the inner workings of the DES algorithm and its key generation process.