Project 0

CSE350: Programming Assignment 2

Parveen 2021079
Shubham Sharma 2021099

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Description

This Assignment Explores the Implementation of DES (*Data Encryption Standard*) and the Verification of Implementation by matching the Deciphered text with Original Plain text.

- DES is a Symmetric-key Algorithm used for Encryption and Decryption.
- It uses multiple sub key generation rounds & Feistel cipher structure.
- For Decryption, the process is essentially the reverse of encryption, using the subkeys in the **reverse** order.
- Also DES produces a strong Avalanche effect that is change in one bit of input produces change in many bits of output.
- However, later on it has been considered insecure due to its small key size (56 bits only).

Constraints & Details

Input block size: 64 bit blocks

Initial key length: 64 bit

Key length after parity drop: 56 bit

Sub key length: 48 bit

No. of rounds : 16

Output block size: 64 bit

Uses **FEISTEL** cipher like structure

Key Components

- **Initial permutation**
  - Rearranges the bits of the input plaintext.
  - Provides diffusion in the Encryption process.
- **Expansion Permutation Box**
  - Increases the amount of data to be processed by each S-box.
  - Provides more diffusion in the encryption process.
- **S-boxes (Substitution Boxes)**
  - Non-linear components that substitute groups of bits from the input.
  - Introduce non-linearity and confusion into the algorithm.
- **P-box (Permutation Box)**
  - Rearranges the bits after the S-box substitution.
  - Further enhances the diffusion of the algorithm.
- **Parity Drop Box**
  - Drops parity bits from the key.
  - Reduces the key size from 64 to 56 bits.
- **Final Permutation Box:**
  - Reorders the bits in the final output ciphertext.
  - Provides a reversible transformation for decryption.

Encryption Process

STEP 1: Key Generation

- Generate round keys by shifting halves of the key.

STEP 2: Encryption Rounds

- Apply initial permutation on plain text.
- Perform expansion permutation and XOR with the key.
- Substitute using S-boxes.
- Apply permutation box and XOR with the left half.
- Repeat for 16 rounds.

STEP 3: Final Permutation

- Apply the final permutation to obtain the Cipher text.

```
def encrypt(left, right, key):
    '''Apply Expansion Permutation and XOR with Key'''

    expansion_result = "".join([right[i - 1]
    |   |   |   |   |   |   |   for i in expansion_permutation_Box])
    xor_result = str(bin(int(expansion_result, 2) ^ int(key, 2)))[2:].zfill(48)

    '''Apply S-Box Substitution'''

    S_Str = ""
    for i in range(0, 8):
    |   S_row = xor_result[(i * 6)] + xor_result[(i * 6) + 5]
    |   S_col = xor_result[(i * 6) + 1] + xor_result[(i * 6) + 2] + \
    |   |   xor_result[(i * 6) + 3] + xor_result[(i * 6) + 4]

    |   S_Str += dec_to_bin(S_Box[i][bin_to_dec(S_row)][bin_to_dec(S_col)])

    '''Apply Permutation Box and XOR with Left Half'''

    return right, str(bin(int(left, 2) ^ int("".join([S_Str[i - 1] for i in P_BOX]), 2)))
```

Decryption Process

STEP 1: Key Generation

- Generate round keys by shifting halves of the key.

STEP 2: Decryption Rounds

- Apply initial permutation to Cipher Text.
- Perform expansion permutation and XOR with the keys in **reverse** order.
- Substitute using S-boxes.
- Apply permutation box and XOR with the left half.
- Repeat for 16 rounds.

STEP 3: Final Permutation

- Apply the final permutation to obtain the Plain Text.

```
def encrypt(left, right, key):
    '''Apply Expansion Permutation and XOR with Key'''

    expansion_result = "".join([right[i - 1]
                                for i in expansion_permutation_Box])
    xor_result = str(bin(int(expansion_result, 2) ^ int(key, 2)))[2:].zfill(48)

    '''Apply S-Box Substitution'''

    S_Str = ""
    for i in range(0, 8):
        S_row = xor_result[(i * 6)] + xor_result[(i * 6) + 5]
        S_col = xor_result[(i * 6) + 1] + xor_result[(i * 6) + 2] + \
            xor_result[(i * 6) + 3] + xor_result[(i * 6) + 4]

        S_Str += dec_to_bin(S_Box[i][bin_to_dec(S_row)][bin_to_dec(S_col)])

    '''Apply Permutation Box and XOR with Left Half'''

    return right, str(bin(int(left, 2) ^ int("".join([S_Str[i - 1] for i in P_BOX]), 2)))
```

Function used for Encryption & Decryption is the same.

Results

- On comparing the final Decipher text with the Original Plain text it has been verified that the code works as per DES algorithm.
- Verifies the Output of Encryption Round 1 is SAME as the Output of Decryption Round 15.
- Verifies the Output of Encryption Round 14 is SAME as the Output of Decryption Round 2.

```
For Plain Text 1:-

Output of Encryption Round 1  is 000000001001010100001001100000011100001010010100110111100101000
Output of Decryption Round 15 is 000000001001010100001001100000011100001010010100110111100101000

Output of Encryption Round 1 is SAME as Output of Decryption Round 15


Output of Encryption Round 14  is 010110001100111100101100100011000101101110001100110010100001010
Output of Decryption Round 2 is 010110001100111100101100100011000101101110001100110010100001010

Output of Encryption Round 14 is SAME as Output of Decryption Round 2


Encryption and Decryption SUCCESSFUL
```