Sameeksha Gupta
22BCP343

Experiment 1

Implementation of SDES: A Simplified Data Encryption Standard

- Objective:
  A program that shows how SDES works for encryption and decryption purposes must be developed. Such a program must take an 8-bit plaintext and a 10-bit key as input and generate subkeys. Then it will produce ciphertext by using the SDES encryption algorithm. At the same time, it has to reverse this process to get back the plaintext from the given ciphertext.

- Code:

```c
#include <stdio.h>
#include <string.h>

int S0[4][4] = {
    {1, 0, 3, 2},
    {3, 2, 1, 0},
    {0, 2, 1, 3},
    {3, 1, 3, 2}
};

int S1[4][4] = {
    {0, 1, 2, 3},
    {2, 0, 1, 3},
    {3, 0, 1, 0},
    {2, 1, 0, 3}
};

int P10[] = {3, 5, 2, 7, 4, 10, 1, 9, 8, 6};
int P8[] = {6, 3, 7, 4, 8, 5, 10, 9};
int IP[] = {2, 6, 3, 1, 4, 8, 5, 7};
int IP_INV[] = {4, 1, 3, 5, 7, 2, 8, 6};
int EP[] = {4, 1, 2, 3, 2, 3, 4, 1};
int P4[] = {2, 4, 3, 1};

void permute(int *input, int *output, int *table, int n) {
    for (int i = 0; i < n; i++) {
        output[i] = input[table[i] - 1];
    }
}
```

Sameeksha Gupta
22BCP343

```c
void left_shift(int *input, int shifts) {
    int temp[5];
    for (int i = 0; i < shifts; i++) {
        for (int j = 0; j < 5; j++) {
            temp[j] = input[j];
        }
        for (int j = 0; j < 4; j++) {
            input[j] = input[j + 1];
        }
        input[4] = temp[0];
    }
}

void generate_keys(int *key, int *K1, int *K2) {
    int permuted[10], LS1[10], LS2[10];

    permute(key, permuted, P10, 10);

    int left[5], right[5];
    for (int i = 0; i < 5; i++) {
        left[i] = permuted[i];
        right[i] = permuted[i + 5];
    }

    left_shift(left, 1);
    left_shift(right, 1);

    for (int i = 0; i < 5; i++) {
        LS1[i] = left[i];
        LS1[i + 5] = right[i];
    }
    permute(LS1, K1, P8, 8);

    left_shift(left, 2);
    left_shift(right, 2);

    for (int i = 0; i < 5; i++) {
        LS2[i] = left[i];
        LS2[i + 5] = right[i];
    }
    permute(LS2, K2, P8, 8);
```

Sameeksha Gupta
22BCP343

```
}

void xor_arrays(int *a, int *b, int *result, int n) {
    for (int i = 0; i < n; i++) {
        result[i] = a[i] ^ b[i];
    }
}

void s_box(int *input, int *output) {
    int row = input[0] * 2 + input[3];
    int col = input[1] * 2 + input[2];
    int s0_value = S0[row][col];

    row = input[4] * 2 + input[7];
    col = input[5] * 2 + input[6];
    int s1_value = S1[row][col];

    output[0] = (s0_value & 2) >> 1;
    output[1] = s0_value & 1;
    output[2] = (s1_value & 2) >> 1;
    output[3] = s1_value & 1;
}

void fk(int *input, int *subkey, int *output) {
    int left[4], right[4], ep_result[8], xor_result[8], sbox_input[8],
sbox_output[4];

    for (int i = 0; i < 4; i++) {
        left[i] = input[i];
        right[i] = input[i + 4];
    }

    permute(right, ep_result, EP, 8);

    xor_arrays(ep_result, subkey, xor_result, 8);

    s_box(xor_result, sbox_output);

    permute(sbox_output, output, P4, 4);

    for (int i = 0; i < 4; i++) {
```

Sameeksha Gupta
22BCP343

```
        output[i] ^= left[i];
        output[i + 4] = right[i];
    }
}

void s_des(int *input, int *K1, int *K2, int *output, int mode) {
    int ip_result[8], fk1_result[8], fk2_result[8];

    permute(input, ip_result, IP, 8);

    fk(ip_result, mode == 0 ? K1 : K2, fk1_result);

    int temp;
    for (int i = 0; i < 4; i++) {
        temp = fk1_result[i];
        fk1_result[i] = fk1_result[i + 4];
        fk1_result[i + 4] = temp;
    }

    fk(fk1_result, mode == 0 ? K2 : K1, fk2_result);

    permute(fk2_result, output, IP_INV, 8);
}

void string_to_binary(char *str, int *binary_array, int length) {
    for (int i = 0; i < length; i++) {
        binary_array[i] = str[i] - '0';
    }
}

int main() {
    char key_input[11], plaintext_input[9];
    int key[10], plaintext[8], ciphertext[8], decrypted[8];
    int K1[8], K2[8];

    printf("Enter 10-bit key (e.g., 1010001010): ");
    scanf("%s", key_input);

    printf("Enter 8-bit plaintext (e.g., 01101010): ");
    scanf("%s", plaintext_input);
```

Sameeksha Gupta
22BCP343

```
    string_to_binary(key_input, key, 10);
    string_to_binary(plaintext_input, plaintext, 8);

    generate_keys(key, K1, K2);

    printf("Key-1: ");
    for (int i = 0; i < 8; i++) printf("%d ", K1[i]);
    printf("\nKey-2: ");
    for (int i = 0; i < 8; i++) printf("%d ", K2[i]);
    printf("\n");

    s_des(plaintext, K1, K2, ciphertext, 0);
    printf("Ciphertext: ");
    for (int i = 0; i < 8; i++) printf("%d ", ciphertext[i]);
    printf("\n");

    s_des(ciphertext, K1, K2, decrypted, 1);
    printf("Decrypted: ");
    for (int i = 0; i < 8; i++) printf("%d ", decrypted[i]);
    printf("\n");

    return 0;
}
```

- Output:

```
PS C:\Users\samee\OneDrive\Documents\PDEU\SEM 6\Lab\Crypto\lab1> .\a.exe
Enter 10-bit key (e.g., 1010001010): 1010000010
Enter 8-bit plaintext (e.g., 01101010): 01110010
Key-1: 1 0 1 0 0 1 0 0
Key-2: 0 1 0 0 0 0 1 1
Ciphertext: 0 1 1 1 0 1 1 1
Decrypted: 0 1 1 1 0 0 1 0
PS C:\Users\samee\OneDrive\Documents\PDEU\SEM 6\Lab\Crypto\lab1>
```