Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2024-25
School Of Computer Science and Engineering
(SCOPE)

Digital Assignment - III
Cryptography and Network Security Lab

Apurva Mishra: 22BCE2791
Date: 9 March, 2025

Contents

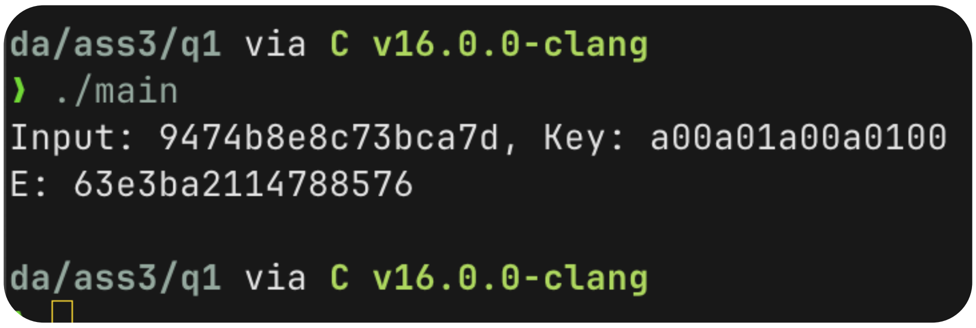
1 DES

1.1 Code

Code 0: main.c

```c
  #include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define LB32_MASK   0x00000001
#define LB64_MASK   0x0000000000000001
#define L64_MASK    0x00000000ffffffff
#define H64_MASK    0xffffffff00000000

/* Initial Permutation Table */
static char IP[] = {
    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
};

/* Inverse Initial Permutation Table */
static char PI[] = {
    40,  8, 48, 16, 56, 24, 64, 32,
    39,  7, 47, 15, 55, 23, 63, 31,
    38,  6, 46, 14, 54, 22, 62, 30,
    37,  5, 45, 13, 53, 21, 61, 29,
    36,  4, 44, 12, 52, 20, 60, 28,
    35,  3, 43, 11, 51, 19, 59, 27,
    34,  2, 42, 10, 50, 18, 58, 26,
    33,  1, 41,  9, 49, 17, 57, 25
};

/*Expansion table */
static char E[] = {
    32,  1,  2,  3,  4,  5,
     4,  5,  6,  7,  8,  9,
     8,  9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
    16, 17, 18, 19, 20, 21,
    20, 21, 22, 23, 24, 25,
    24, 25, 26, 27, 28, 29,
    28, 29, 30, 31, 32,  1
};

/* Post S-Box permutation */
static char P[] = {
    16,  7, 20, 21,
    29, 12, 28, 17,
     1, 15, 23, 26,
     5, 18, 31, 10,
     2,  8, 24, 14,
    32, 27,  3,  9,
    19, 13, 30,  6,
    22, 11,  4, 25
};

/* The S-Box tables */
static char S[8][64] = {{
    /* S1 */
    14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7,
     0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8,
     4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0,
    15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13
},{
    /* S2 */
    15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5, 10,
     3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,  5,
     0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2, 15,
    13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,  9
},{
    /* S3 */
    10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8,
    13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1,
    13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7,
     1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2, 12
},{
    /* S4 */
     7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4, 15,
    13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,  9,
    10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,  4,
     3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2, 14
},{
    /* S5 */
     2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,  9,
    14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6,
     4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14,
    11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,  3
},{
    /* S6 */
    12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5, 11,
    10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,  8,
     9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,  6,
     4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8, 13
},{
    /* S7 */
     4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,  1,
    13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,  6,
     1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,  2,
     6, 11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3, 12
},{
    /* S8 */
    13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,  7,
     1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,  2,
     7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,  8,
     2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6, 11
}};

/* Permuted Choice 1 Table */
static char PC1[] = {
    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
};

/* Permuted Choice 2 Table */
static char PC2[] = {
    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
};

/* Iteration Shift Array */
static char iteration_shift[] = {
 /* 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16 */
    1,  1,  2,  2,  2,  2,  2,  2,  1,  2,  2,  2,  2,  2,  2,  1
};


uint64_t des(uint64_t input, uint64_t key) {

    int i, j;

    /* 8 bits */
    char row, column;

    /* 28 bits */
    uint32_t C                  = 0;
    uint32_t D                  = 0;

    /* 32 bits */
    uint32_t L                  = 0;
    uint32_t R                  = 0;
    uint32_t s_output           = 0;
    uint32_t f_function_res     = 0;
    uint32_t temp               = 0;

    /* 48 bits */
    uint64_t sub_key[16]        = {0};
    uint64_t s_input            = 0;

    /* 56 bits */
    uint64_t permuted_choice_1  = 0;
    uint64_t permuted_choice_2  = 0;

    /* 64 bits */
    uint64_t init_perm_res      = 0;
    uint64_t inv_init_perm_res  = 0;
    uint64_t pre_output         = 0;

    /* Init Key */

    /* initial key permutation 64 -> 56 */
    for (i = 0; i < 56; i++) {

        permuted_choice_1 <<= 1;
        permuted_choice_1 |= (key >> (64-PC1[i])) & LB64_MASK;

    }

    /* initial split C_(0) and D_(0) */
    C = (uint32_t) ((permuted_choice_1 >> 28) & 0x000000000fffffff);
    D = (uint32_t) (permuted_choice_1 & 0x000000000fffffff);

    /* Calculation of the 16 keys */
    for (i = 0; i< 16; i++) {

        /* shifting C_(i) and D_(i) */
        for (j = 0; j < iteration_shift[i]; j++) {

            C = 0x0fffffff & (C << 1) | 0x00000001 & (C >> 27);
            D = 0x0fffffff & (D << 1) | 0x00000001 & (D >> 27);

        }

        /* combine C and D together */
        permuted_choice_2 = 0;
        permuted_choice_2 = (((uint64_t) C) << 28) | (uint64_t) D ;

        sub_key[i] = 0;

        /* same as initil permutation without tmp variable */
        for (j = 0; j < 48; j++) {

            sub_key[i] <<= 1;
            sub_key[i] |= (permuted_choice_2 >> (56-PC2[j])) & LB64_MASK;

        }

    }


    /* Init Input */
    /* initial input permutation */
    for (i = 0; i < 64; i++) {
        uint64_t tmp = input >> (64 - IP[i]);
        tmp = tmp & LB64_MASK;

        init_perm_res <<= 1;
        init_perm_res |= tmp;
    }

    /* Initial key split: C_(0) and D_(0)*/
    L = (uint32_t) (init_perm_res >> 32) & L64_MASK;
    R = (uint32_t) init_perm_res & L64_MASK;


    /* rounds */
    for (i = 0; i < 16; i++) {

        s_input = 0;

        /* start of round fn */
        /* expand R from 32 -> 48 */
        for (j = 0; j< 48; j++) {

            s_input <<= 1;
            s_input |= (uint64_t) ((R >> (32-E[j])) & LB32_MASK);

        }

        // xor R and key
        s_input = s_input ^ sub_key[i];


        /* S-Box Tables */
        for (j = 0; j < 8; j++) {
            // 00 00 RCCC CR00 00 00 00 00 00 s_input
            // 00 00 1000 0100 00 00 00 00 00 row mask
            // 00 00 0111 1000 00 00 00 00 00 column mask

            row = (char) ((s_input & (0x0000840000000000 >> 6*j)) >> 42-6*j);
            row = (row >> 4) | row & 0x01;

            column = (char) ((s_input & (0x0000780000000000 >> 6*j)) >> 43-6*j);

            s_output <<= 4;
            s_output |= (uint32_t) (S[j][16*row + column] & 0x0f);

        }

        f_function_res = 0;

        /* final permutation */
        for (j = 0; j < 32; j++) {

            f_function_res <<= 1;
            f_function_res |= (s_output >> (32 - P[j])) & LB32_MASK;

        }

        /* final swap */
        temp = R;
        R = L ^ f_function_res;
        L = temp;

    }

    pre_output = (((uint64_t) R) << 32) | (uint64_t) L;

    /* inverse initial permutation */
    for (i = 0; i < 64; i++) {

        inv_init_perm_res <<= 1;
        inv_init_perm_res |= (pre_output >> (64-PI[i])) & LB64_MASK;

    }

    return inv_init_perm_res;

}

int main(int argc, const char * argv[]) {

    int i;

    uint64_t input = 0x9474B8E8C73BCA7D;
    uint64_t key = 0x000AB00A0B00A0A0;

    printf ("Input: %016llx, Key: %llu\n", input, key);

    uint64_t result = des(input, key);
    printf ("E: %016llx\n", result);

    exit(0);
}

```

1.2 Output



```
da/ass3/q1 via C v16.0.0-clang
❯ ./main
Input: 9474b8e8c73bca7d, Key: a00a01a00a0100
E: 63e3ba2114788576

da/ass3/q1 via C v16.0.0-clang
```