Q1 Teamname

0 Points

NULL

Q2 Commands

10 Points

List the commands used in the game to reach the ciphertext.

go, dive, dive, back, pull, back, back, go, wave, back, thrnxxtzy, read, 360852885036840078603672 5, c, read, password

Q3 Cryptosystem

5 Points

What cryptosystem was used at this level? Please be precise.

6-round DES

Q4 Analysis

80 Points

Knowing which cryptosystem has been used at this level, give a detailed description of the cryptanalysis used to figure out the password. (Explain in less than 150 lines and use Latex wherever required. If your solution is not readable, you will lose marks. If necessary, the file upload option in this question must be used TO SHARE IMAGES ONLY.)

Assuming that the cryptosystem used is 6-round DES we did the following to find the key:

Our approach was to do a chosen plain text attack using differential cryptanalysis. Since we need to break 6-round DES, we need a 4-round characteristic. We used the following characteristic which is mentioned in the lecture slides.

$$\left(405C\overline{0}, 0400\overline{0}, \frac{1}{4}, 0400\overline{0}, 0054\overline{0}, \frac{5}{128}, 0054\overline{0}, \overline{00}, 1, \overline{00}, 0054\overline{0}\ \frac{5}{128}, 00\right.$$

The probability of the characteristic is 0.00038 which means that we need atleast $20/0.00038 \approx 53000$ pairs of plaintext pairs with xor equal to $405C\overline{0}0400\overline{0}$ after applying the initial permutation. So we need plain text pairs with xor equal to $IP^{-1}(405C\overline{0}0400\overline{0}) = 0000901010005000$ so that after applying IP their xor would be equal to the required value.

The hint mentions that 2 letters are represented by 1 byte, so only 256 distinct pairs of letters can be present in the ciphertext. To find those pairs, we queried using random inputs, analyzed the outputs, and found that the 256 pairs are all possible pairs of 2 letters where each letter comes from 'f' to 'u' in the alphabet using the program "analyse.cpp". Thus, We assumed that 'f' is represented as 0000, 'g' as 0001, ... , 'u' as 1111 and generated 100000 input pairs with above required xor and queried their outputs using the program
"gen_input_output_pairs.cpp".

We converted the outputs into their bit representation assuming the bit representation of each letter as stated above and applied the Initial Permutation to undo the inverse IP which would be done at the end of DES and also swapped the Left and right parts to get the output of Round 6 of DES using "process_outputs.cpp".

To find the sixth round key, we did the following:
Firstly, we define some notation which we use later
- L_iR_i denotes the output of ith round of DES.
- Let $E\left(R_5\right) = \alpha_1\alpha_2\cdots\alpha_8 = \alpha$ and $E\left(R_5'\right) = \alpha_1'\alpha_2'\cdots\alpha_8' = \alpha'$ with $|\alpha_i| = 6 = |\alpha_i'|$
- R_5 and R_5' are right-halves of output of fifth round on the plaintexts L_0R_0 and $L_0'R_0'$ where $L_0 \oplus L_0' = 405C\overline{0}$ and $R_0 \oplus R_0' = 0400\overline{0}$.
- Let $\beta_i = \alpha_i \oplus k_{6,i}$ and $\beta_i' = \alpha_i' \oplus k_{6,i}, |\beta_i| = 6 = |\beta_i'|$.
- $k_6 = k_{6,1}k_{6,2}\cdots k_{6,8}$
- Let $\gamma_i = S_i\left(\beta_i\right)$ and $\gamma_i' = S_i\left(\beta_i'\right), |\gamma_i| = 4 = |\gamma_i'|$

For a given pair of plaintexts $L_0R_0, L_0'R_0'$ with xor value $405C\overline{0}0400\overline{0}$, we know the output of 6th round of DES i.e $L_6R_6, L_6'R_6'$, Since $R_5 = L_6$,we also know the output of expansion in round 6 $E(R_5), E(R_5')$. And since we know the value of $L_5 \oplus L_5' = R_4 \oplus R_4' = 0400\overline{0}$ with probability 0.00038, we know the value of $\gamma = S(\beta) \oplus S(\beta') = P^{-1}(L_5 \oplus L_5' \oplus R_6 \oplus R_6')$ with probability 0.00038.

- Thus, We know $\alpha_i, \alpha_i', \beta_i \oplus \beta_i' = \alpha_i \oplus \alpha_i'$, and a value of γ such that $S(\beta) \oplus S(\beta') = \gamma$ with probability 0.000381.

For each $1 \leq i \leq 8$, we generated all possible (β_i, β_i') such that $\beta_i \oplus \beta_i' = \alpha_i \oplus \alpha_i'$ and $S(\beta_i) \oplus S(\beta_i') = \gamma_i$ and added the corresponding key $k = \alpha_i \oplus \beta_i$ to a hashmap containing list of possible $k_{6,i}$ values. We did this for all 10000 pairs and took the key with highest frequency in the hashmap as $k_{6,i}$. The code for this can be found in "find_r6_key.cpp".

The keys obtained are 101101,110011,100101,001011,110111,000011,010010,100011.

Here are the frequencies of most frequent, second most frequent element in the hashmap for each $1 \leq i \leq 8$.

For $i = 1$ 8281, 6682
For $i = 2$ 8667, 6682
For $i = 3$ 6418, 6417
For $i = 4$ 6479, 6447
For $i = 5$ 9221, 6821
For $i = 6$ 8741, 6613
For $i = 7$ 8422, 6674
For $i = 8$ 8719, 6872

As we can see, the first most frequent element and second element do not differ much for $i = 3$ and $i = 4$. So we discarded $k_{6,3}$ and $k_{6,4}$ obtained from the above analysis and mapped the other bits to their positions in the main key using key scheduling algorithm and then used brute force to find them along with the other 8 unknown bits. We used a known plaintext ciphertext pair "pjjjstpmigntlltf", "snmfqkhtsjoinofl" and brute forced all 2^{20} possibilities to find the remaining 20 bits.The code for this can be found in "find_key.cpp".

Using the above analysis, the key (after applying PC-1) was found to be $0110111001011110011110110000111001011010011100110010000 1$.

Using the hint given on the screen, we got that the encrypted password is "ountnlqktqortppnkkqqnrhipifmiigt". Since the password is 128 bits long we divided it into two blocks of 64 bits each and decrypted each of them using the DES decryption algorithm as we know the key. We converted the decrypted password to alphabetic representation assuming that each character is 4 bits but the formed password did not work. So we tried

assuming each letter is 8 bits (like the usual ASCII representation) and got "mkpnizcefq000000" as output which also did not work. After some trial and error, we observed that removing zeros at the end and entering "mkpnizcefq" works which also makes sense since the zeros at the end might be added just to make the length of text multiple of the block size. The code for this portion can be found in "find_key.cpp".

Thus we found that the password is "mkpnizcefq".

No files uploaded

Q5 Password

5 Points

What was the final command used to clear this level?

mkpnizcefq

Q6 Codes

0 Points

Unlike previous assignments, this time it is mandatory that you upload the codes used in the cryptanalysis. If you fail to do so, you will be given 0 marks for the entire assignment.

analyse.cpp Download

```cpp
#include <bits/stdc++.h>
using namespace std;

#define endl '\n'

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
int getRand(int l, int r)
{
    uniform_int_distribution<int> uid(l, r);
    return uid(rng);
}

void getinputs()
```

```
{
    ofstream fout("test_inputs.txt");

    int L = 1000;
    for (int i = 0; i < L; i++)
    {
        uint64_t first = 0;
        string s = "";
        for (int j = 0; j < 8; j++)
        {
            int k = getRand(0, 255);
            first = (first << 4) + k;
            s += char(k);
        }
        fout << s << endl;
    }
}

void getoutputs()
{
    ifstream fin("test_inputs.txt");
    ofstream fout("test_cmds.txt");

    fout << "NULL" << endl;
    fout << "foobar268" << endl;
    fout << 4 << endl;

    fout << "read" << endl;

    string s;
    while (fin >> s)
    {
        fout << s << endl;
        fout << 'c' << endl;
    }

    fout << "back" << endl;
    fout << "exit" << endl;

    fout.close();
    fin.close();

    system("ssh student@65.0.124.36 < test_cmds.txt > out");
    system("grep --no-group-separator -A 1 \"Slowly, a new text starts appearing on the screen. It reads ...\" out | grep --no-group-separator -v \"Slowly, a new text starts appearing on the screen. It reads ...\" | tr -d \"\\t\" > test_outputs.txt");
    system("rm -rf out test_cmds.txt");
}

void analyze()
{
```

```
    ifstream fin("test_outputs.txt");
    map<pair<char,char>, int> cnt;
    string s;
    while (fin >> s)
    {
        int n = s.size();
        for (int i = 0; i < n; i += 2)
        {
            cnt[make_pair(s[i], s[i + 1])]++;
        }
    }
    for (auto [c, _] : cnt)
    {
        cout << c.first << ' ' << c.second << endl;
    }
    cout << endl;
}

int main()
{

    getinputs();
    getoutputs();
    analyze();

    return 0;
}

```

gen_input_output_pairs.cpp

Download

```
#include <bits/stdc++.h>
using namespace std;

#define endl '\n'

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
int getRand(int l, int r)
{
    uniform_int_distribution<int> uid(l, r);
    return uid(rng);
}

void getinputs()
{
    uint64_t diff = 0x405c000004000000;

    ofstream fout("inputs.txt");
    int 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};

    uint64_t rdiff = 0;
    for (int i = 0; i < 64; i++)
    {
        uint64_t cur = ((diff >> (63 - i)) & 1);
        int to = IP[i];
        rdiff |= (cur << (64 - to));
    }

    cout << hex << setw(16) << setfill('0') << rdiff << endl;

    int L = 1e5;
    for (int i = 0; i < L; i++)
    {
        uint64_t first = 0;
        string s = "";
        for (int j = 0; j < 16; j++)
        {
            int k = getRand(0, 15);
            first = (first << 4) + k;
            s += char('f' + k);
        }
        fout << s << endl;
        s = "";
        uint64_t second = (first ^ rdiff);
        for (int j = 0; j < 16; j++)
        {
            int num = 0;
            for (int k = 0; k < 4; k++)
            {
                int at = 4 * j + k;
                int cur = ((second >> (63 - at)) & 1);
                num += (cur << (3 - k));
            }
            s += char('f' + num);
        }
        fout << s << endl;
    }
}

void getoutputs()
{
    ifstream fin("inputs.txt");
    ofstream fout("cmds.txt");

    fout << "NULL" << endl;
```

```
    fout << "foobar268" << endl;
    fout << 4 << endl;

    fout << "read" << endl;

    string s;
    while (fin >> s)
    {
        fout << s << endl;
        fout << 'c' << endl;
    }

    fout << "back" << endl;
    fout << "exit" << endl;

    fout.close();
    fin.close();

    system("ssh student@65.0.124.36 < cmds.txt > out");
    system("grep --no-group-separator -A 1 \"Slowly, a new text starts appearing on the screen. It reads ...\" out | grep --no-group-separator -v \"Slowly, a new text starts appearing on the screen. It reads ...\" | tr -d \"\\t\" > outputs.txt");
    system("rm -rf out cmds.txt");
}

int main()
{

    getinputs();
    getoutputs();

    return 0;
}

```

process_outputs.cpp

Download

```
#include <bits/stdc++.h>

using namespace std;

using b64 = bitset<64>;
using b32 = bitset<32>;
using b48 = bitset<48>;
using b56 = bitset<56>;
using b28 = bitset<28>;

int 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};

int Inv_IP[] = {
    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};

b64 apply_ip(b64 in, bool inv = false)
{
    b64 out;
    for (int i = 63; i >= 0; i--)
    {
        int j = (inv ? Inv_IP[63 - i] : IP[63 - i]) - 1;
        out[i] = in[63 - j];
    }
    return out;
}

pair<b32, b32> get_LR(b64 in)
{
    b32 L, R;
    for (int i = 63, j = 31; i >= 32; i--, j--)
    {
        L[j] = in[i];
        R[j] = in[i - 32];
    }
    return make_pair(L, R);
}
b64 join(b32 L, b32 R)
{
    b64 out;
    for (int i = 31; i >= 0; i--)
    {
        out[i + 32] = L[i];
        out[i] = R[i];
    }
    return out;
}

int main()
{

    ifstream fin("outputs.txt");
```

```
    ofstream fout("outbits.txt");

    string s;
    while (fin >> s)
    {
        assert((int)s.length() == 16);
        b64 bits = 0;
        for (int i = 0; i < 16; i++)
        {
            char c = s[i];
            int num = c - 'f';
            assert(num < 16 and num >= 0);
            bits = (bits << 4) | b64(num);
        }
        b64 actual = apply_ip(bits);
        auto [R, L] = get_LR(actual);

        fout << L << R << endl;
    }

    return 0;
}
```

find_r6_key.cpp

Download

```
#include <bits/stdc++.h>

using namespace std;

using b64 = bitset<64>;
using b32 = bitset<32>;
using b48 = bitset<48>;
using b56 = bitset<56>;
using b28 = bitset<28>;

int 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};

int Inv_IP[] = {
    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};

b64 apply_ip(b64 in, bool inv = false)
{
    b64 out;
    for (int i = 63; i >= 0; i--)
    {
        int j = (inv ? Inv_IP[63 - i] : IP[63 - i]) - 1;
        out[i] = in[63 - j];
    }
    return out;
}

pair<b32, b32> get_LR(b64 in)
{
    b32 L, R;
    for (int i = 63, j = 31; i >= 32; i--, j--)
    {
        L[j] = in[i];
        R[j] = in[i - 32];
    }
    return make_pair(L, R);
}

int 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};

b48 apply_E(b32 in)
{
    b48 out;
    for (int i = 47; i >= 0; i--)
    {
        int j = E[47 - i] - 1;
        out[i] = in[31 - j];
    }
    return out;
}

int S1[4][16] = {
    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};

```

```
int S2[4][16] = {
    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};

int S3[4][16] = {
    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};

int S4[4][16] = {
    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};

int S5[4][16] = {
    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};

int S6[4][16] = {
    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};

int S7[4][16] = {
    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};

int S8[4][16] = {
    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};

b32 apply_S(b48 in)
{
    b32 out;
    int i = 47, j = 31;
    for (auto S : {S1, S2, S3, S4, S5, S6, S7, S8})
    {
        int row = 0, col = 0;
        for (int k = i - 1; k > i - 5; k--)
        {
            col = col * 2 + in[k];
```

```
        }
        row = in[i] * 2 + in[i - 5];

        bitset<4> num = S[row][col];
        for (int k = j; k >= j - 3; k--)
        {
            out[k] = num[3 - (j - k)];
        }
        i -= 6;
        j -= 4;
    }
    return out;
}

int 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};

int Inv_P[32];

void calc_Inv_P()
{
    for (int i = 0; i < 32; i++)
    {
        int to = P[i] - 1;
        Inv_P[to] = i + 1;
    }
}

b32 apply_P(b32 in, bool inv = false)
{
    b32 out;
    for (int i = 31; i >= 0; i--)
    {
        int j = (inv ? Inv_P[31 - i] : P[31 - i]) - 1;
        out[i] = in[31 - j];
    }
    return out;
}

using b4 = bitset<4>;
using b6 = bitset<6>;

b4 calc(b6 in, int i)
{
    b4 out;
    int cur = 0;
    for (auto S : {S1, S2, S3, S4, S5, S6, S7, S8})
    {
        if (i == cur)
```

```
        {
            int row = 0, col = 0;
            for (int k = 4; k > 0; k--)
            {
                col = col * 2 + in[k];
            }
            row = in[5] * 2 + in[0];

            bitset<4> num = S[row][col];
            for (int k = 3; k >= 0; k--)
            {
                out[k] = num[k];
            }
            return out;
        }
        cur = cur + 1;
    }
    assert(false);
    return out;
}
#define endl '\n'

int main()
{
    ifstream fin("outbits.txt");
    calc_Inv_P();

    int N = 1e5;

    unordered_map<bitset<6>, int> maybe[8];

    for (int _ = 0; _ < N; _++)
    {
        bitset<32> out1L, out1R, out2L, out2R;
        fin >> out1L >> out1R >> out2L >> out2R;

        // outputs after expansion in the last round
        bitset<48> out1E = apply_E(out1L);
        bitset<48> out2E = apply_E(out2L);
        bitset<48> inxorS = (out1E ^ out2E);

        // cout << inxorS << endl;

        bitset<32> xorL = bitset<32>(0x04000000);
        bitset<32> xorP = (out1R ^ out2R ^ xorL);
        bitset<32> outxorS = apply_P(xorP, true);

        for (int i = 0; i < 8; i++)
        {
            int st = i * 6;
            for (int j = 0; j < (1 << 6); j++)
            {
```

```
                bitset<6> in1 = j;
                bitset<6> in2 = 0;
                for (int k = st; k - st < 6; k++)
                {
                    in2[5 - k + st] = (in1[5 - k + st] ^ inxorS[47 - k]);
                }
                bitset<4> out1 = calc(in1, i);
                bitset<4> out2 = calc(in2, i);
                bitset<4> got = (out1 ^ out2);
                bool ok = 1;
                for (int k = i * 4; k - i * 4 < 4; k++)
                {
                    ok &= (outxorS[31 - k] == got[3 - k + i * 4]);
                }

                if (ok)
                {
                    bitset<6> cand;
                    for (int k = st; k - st < 6; k++)
                    {
                        cand[5 - k + st] = in1[5 - k + st] ^ out1E[47 - k];
                    }
                    maybe[i][cand]++;
                }
            }
        }
    }

    bitset<48> key = 0;

    for (int i = 0; i < 8; i++)
    {
        int cur_mx = 0;
        bitset<6> ans;
        for (auto [bs, cnt] : maybe[i])
        {
            if (cnt > cur_mx)
            {
                cur_mx = cnt;
                ans = bs;
            }
        }
        key = (key << 6);
        for (int j = 0; j < 6; j++)
        {
            key[j] = ans[j];
        }
    }

    cout << key << endl;
```

```

    return 0;
}
```

find_key.cpp Download

```cpp
#include <bits/stdc++.h>

using namespace std;

using b64 = bitset<64>;
using b32 = bitset<32>;
using b48 = bitset<48>;
using b56 = bitset<56>;
using b28 = bitset<28>;

int 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};

int Inv_IP[] = {
    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};

b64 apply_ip(b64 in, bool inv = false)
{
    b64 out;
    for (int i = 63; i >= 0; i--)
    {
        int j = (inv ? Inv_IP[63 - i] : IP[63 - i]) - 1;
        out[i] = in[63 - j];
    }
    return out;
}

pair<b32, b32> get_LR(b64 in)
{
    b32 L, R;
    for (int i = 63, j = 31; i >= 32; i--, j--)
    {
```

```
        L[j] = in[i];
        R[j] = in[i - 32];
    }
    return make_pair(L, R);
}

int 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};

b48 apply_E(b32 in)
{
    b48 out;
    for (int i = 47; i >= 0; i--)
    {
        int j = E[47 - i] - 1;
        out[i] = in[31 - j];
    }
    return out;
}

int S1[4][16] = {
    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};

int S2[4][16] = {
    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};

int S3[4][16] = {
    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};

int S4[4][16] = {
    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};

int S5[4][16] = {
```

```
    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};

int S6[4][16] = {
    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};

int S7[4][16] = {
    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};

int S8[4][16] = {
    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};

b32 apply_S(b48 in)
{
    b32 out;
    int i = 47, j = 31;
    for (auto S : {S1, S2, S3, S4, S5, S6, S7, S8})
    {
        int row = 0, col = 0;
        for (int k = i - 1; k > i - 5; k--)
        {
            col = col * 2 + in[k];
        }
        row = in[i] * 2 + in[i - 5];

        bitset<4> num = S[row][col];
        for (int k = j; k >= j - 3; k--)
        {
            out[k] = num[3 - (j - k)];
        }
        i -= 6;
        j -= 4;
    }
    return out;
}

int 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};
```

```
int Inv_P[32];

void calc_Inv_P()
{
    for (int i = 0; i < 32; i++)
    {
        int to = P[i] - 1;
        Inv_P[to] = i + 1;
    }
}

b32 apply_P(b32 in, bool inv = false)
{
    b32 out;
    for (int i = 31; i >= 0; i--)
    {
        int j = (inv ? Inv_P[31 - i] : P[31 - i]) - 1;
        out[i] = in[31 - j];
    }
    return out;
}

using v64 = array<int, 64>;
using v32 = array<int, 32>;
using v48 = array<int, 48>;
using v56 = array<int, 56>;
using v28 = array<int, 28>;

int 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};

unsigned short shifts[] = {
    1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

int 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};

```

```
vector<v48> keys(16);

void get_keys()
{
    v56 key;
    for (int i = 55; i >= 0; i--)
    {
        key[i] = 56 - i;
    }

    v28 C, D;
    for (int i = 55, j = 27; i >= 28; i--, j--)
    {
        C[j] = key[i];
        D[j] = key[i - 28];
    }

    for (int r = 0; r < 16; r++)
    {
        v28 rC, rD;
        int s = shifts[r];
        for (int i = 27; i >= 0; i--)
        {
            int to = (i + s) % 28;
            rC[to] = C[i];
            rD[to] = D[i];
        }
        C = rC, D = rD;

        for (int i = 47; i >= 0; i--)
        {
            int j = PC2[47 - i] - 1;
            j = 55 - j;
            if (j > 27)
            {
                keys[r][i] = C[j - 28];
            }
            else
            {
                keys[r][i] = D[j];
            }
        }
    }
}

void print(v48 key)
{
    for (int i = 47; i >= 0; i -= 6)
    {
        int num = 0;
        for (int j = i; j >= i - 5; j--)
        {
```

```
            num = num * 2 + key[j];
        }
        cout << setw(2) << setfill('0') << hex << num;
    }
    cout << " ";
}

b64 join(b32 L, b32 R)
{
    b64 out;
    for (int i = 31; i >= 0; i--)
    {
        out[i + 32] = L[i];
        out[i] = R[i];
    }
    return out;
}

b64 get(string s)
{
    assert(s.length() == 16);
    b64 ans = 0;
    for (int i = 0; i < 16; i++)
    {
        int foo = s[i] - 'f';
        ans = (ans << 4) | b64(foo);
    }
    return ans;
}

vector<b48> rkeys(16);
void get_keys2(vector<int> key)
{
    for (int r = 0; r < 16; r++)
    {
        for (int i = 47; i >= 0; i--)
        {
            int j = keys[r][i] - 1;
            rkeys[r][i] = key[55 - j];
        }
    }
}

int main()
{
    get_keys();
    b48 key_r6 = b48("101101110011100101001011110111000011010010100011");

    vector<int> key(56, -1);
    for (int i = 47, cur = 0; i >= 0; i -= 6, cur++)
    {
```

```
        if (cur == 2 or cur == 3)
            continue;
        for (int j = i; j >= i - 5; j--)
        {
            int k = keys[5][j] - 1;
            key[55 - k] = key_r6[j];
        }
    }

    auto doDES = [&](b64 in, bool inv = false) {
        b64 LR = apply_ip(in);
        auto [L, R] = get_LR(LR);
        for (int r = 0; r < 6; r++)
        {
            auto ER = apply_E(R);
            ER ^= (inv ? rkeys[5 - r] : rkeys[r]);
            auto SR = apply_S(ER);
            auto PR = apply_P(SR);
            auto nL = R, nR = L ^ PR;
            L = nL, R = nR;
        }
        auto O = join(R, L);
        auto out = apply_ip(O, true);
        return out;
    };

    vector<int> pos;
    int cnt = 0;
    for (int i = 55; i >= 0; i--)
    {
        if (key[i] == -1)
            pos.push_back(i);
        cnt += (key[i] == -1);
    }

    calc_Inv_P();

    b64 in = get("pjjjstpmigntlltf");
    b64 out = get("snmfqkhtsjoinofl");

    for (int i = 0; i < (1 << cnt); i++)
    {
        for (int j = 0; j < cnt; j++)
        {
            if ((i & (1 << j)) != 0)
            {
                key[pos[j]] = 1;
            }
            else
                key[pos[j]] = 0;
        }
        get_keys2(key);
```

```
        if (doDES(in) == out)
        {
            cout << "Found the key !!" << endl;
            for (int k = 55; k >= 0; k--)
            {
                cout << key[k];
            }
            cout << endl;
            break;
        }
    }

    b64 in21 = get("ountnlqktqortppn");
    b64 in22 = get("kkqqnrhipifmiigt");

    for (auto in2 : {in21, in22})
    {
        auto out2 = doDES(in2, true);
        for (int i = 63; i >= 0; i -= 8)
        {
            int num = 0;
            for (int j = i; j >= i - 7; j--)
            {
                num = num * 2 + out2[j];
            }
            cout << char(num);
        }
    }
    cout << endl;

    return 0;
}
```

Assignment 4

UNGRADED

GROUP

AJAY PRAJAPATI
A5 - SURYADEVARA SAI KRISHNA
A11 - GARIMELLA MOHAN RAGHU
View or edit group

TOTAL POINTS

- / 100 pts

QUESTION 1

Teamname 0 pts

QUESTION 2

Commands 10 pts

QUESTION 3

Cryptosystem 5 pts

QUESTION 4

Analysis 80 pts

QUESTION 5

Password 5 pts

QUESTION 6

Codes 0 pts