Q1 Teamname

0 Points

NULL

Q2 Commands

5 Points

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

go,go,go,go,go,give,read

Q3 Analysis

50 Points

Give a detailed description of the cryptanalysis used to figure out the password. (Explain in less than 100 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.)

1. Understanding the encryption algorithm

The problem statement was to decrypt the encrypted message which was encrypted using a procedure which is similar to SHA-3 (which was given to us as C code). We analyzed the encryption algorithm and obtained the following inferences about its working:

(a) The encryption algorithm pads the message with zeroes until its length becomes 576 bits.
(b) The algorithm maintains a state (which is 5*5*64 array) which gets updated as we proceed through the algorithm.
(c) The state is initialized such that the first 576 values in the array (in row major order) are initialized to the corresponding bits of the padded message and the remaining bits are initialized to zero.
(d) In each round, The algorithm applies theta, Pi, Chi operation once on the current state and there are 24 such rounds.

The code for theta,Pi,chi operations are given below

```
//theta operation
for(i = 0; i < 5; ++i){
        for(k = 0; k < 64; ++k){
                column_parity[i][k] = 0;
                for(j = 0; j < 5; ++j)
                        column_parity[i][k] ^= state[i][j][k];
        }
}

for(i = 0; i < 5; ++i){
        for(j = 0; j < 5; ++j){
            for(k = 0; k < 64; ++k){
                state[i][j][k] ^= column_parity[(i+4)%5][k] ^ column_parity[(i+1)%5][k];
                tempstate[i][j][k] = state[i][j][k];
            }
        }
}
//pi operation
for(i = 0; i < 5; ++i)
    for(j = 0; j < 5; ++j)
        for(k = 0; k < 64; ++k)
            state[j][((2 * i) + (3 * j)) % 5][k] = tempstate[i][j][k];

//chi operation
for(i = 0; i < 5; ++i)
    for(j = 0; j < 5; ++j)
        for(k = 0; k < 64; ++k)
            tempstate[i][j][k] = state[i][j][k];

for(i = 0; i < 5; ++i)
    for(j = 0; j < 5; ++j)
        for(k = 0; k < 64; ++k)
            state[i][j][k] = tempstate[i][j][k] ^ (~tempstate[i][(j+1)%5][k] & tempstate[i][(j+2)%5][k]);
```

(e) The encrypted message is obtained by taking the first 512 bits of the final state (in row major order).

2. Breaking the algorithm

The main observation which enabled us to break the encryption is the following:

Let $state, newstate$ be the states before and after applying one round of theta,pi and chi operations.

For all $0 \leq i \leq 4, 0 \leq j \leq 4, 0 \leq k \leq 63$ the value of $newstate[i][j][k]$ only depends on the set of values $S = \{state[i'][j'][k] : 0 \leq i' \leq 4, 0 \leq j' \leq 4\}$. This can be seen by carefully observing the code given above(k does not change in any of the operations while updating state).

The observation essentially implies that to calculate the value of a cell (i,j,k) in the final state after 24 rounds, we only need to know the values of cells with same value of the third coordinate in the initial state.

The encrypted password was given to be the following
6A6460690000000001018004666168EA6B6FE5ED666168EA010B8584666168EA
000A0580000000000010B8584666168EA6A6E65E9000000006A64606900000000

So our goal is to find the initial state given the first 512 bits (above) of the final state (after applying 24 rounds). Also, the password was given to be no more than 16 characters, so we only need to find the first 128 bits of the initial state (and the rest are all zeroes).

Our strategy was to use brute force combined with above observation to find the password. Using the above observation we can do the following:

For each k from 0 to 63:

Let $S_k = \{(i, j, k) : 0 \leq i \leq 4, 0 \leq j \leq 4, i * 64 * 5 + j * 64 + k < 128\}$ which is the set of all cells such that the third coordinate equal to k and is within the first 128 cells.

It can be clearly seen that $|S_k| = 2$ owing to the way in which the state is initialized using the plain text.

we try all $2^{|S_k|}$ possibilities for the values of these cells in the initial state and assign the one which gives the correct values for all the cells $(i', j', k), 0 \leq i' \leq 4, 0 \leq j' \leq 4$ in the final state(which is given above) after 24 rounds.

The correctness of the above algorithm directly follows from the observation. Using the above procedure, we obtained the password "mjpkfhauegjy" which cleared the level.

The code for the above can be seen in the code section below.

No files uploaded

Q4 Password

25 Points

What was the final command used to clear this level?

mjpkfhauegjy

Q5 Codes

0 Points

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.

▾ solver.cpp Download

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

void prepare(string &s)
{
    string ans = "";
    for (int i = 0; i < s.length(); i++)
    {
        int x;
        if (s[i] >= '0' and s[i] <= '9')
        {
            x = s[i] - '0';
        }
        if (s[i] >= 'A' and s[i] <= 'F')
        {
            x = s[i] - 'A' + 10;
        }

        for (int j = 0; j < 4; j++)
        {
            if ((x & (1 << j)) != 0)
                ans += '1';
            else
                ans += '0';
        }
    }
    s = ans;
}

```

```cpp
int main()
{
    string s =
"6A6460690000000001018004666168EA6B6FE5ED666168EA010B8584666168EA000A0580000000000

    prepare(s);
    uint64_t final_state[5][5][64];

    assert(int(s.size()) == 512);

    for (int i = 0, cur = 0; cur < 512; i++)
    {
        for (int j = 0; j < 5 and cur < 512; j++)
        {
            for (int k = 0; k < 64 and cur < 512; k++)
            {
                final_state[i][j][k] = s[cur++] - '0';
            }
        }
    }

    uint64_t state[5][5][64], ans[5][5][64];
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 5; j++)
            for (int k = 0; k < 64; k++)
                state[i][j][k] = ans[i][j][k] = 0;

    vector<int> adj[64];
    for (int i = 0, cur = 0; cur < 128; i++)
    {
        for (int j = 0; j < 5 and cur < 128; j++)
        {
            for (int k = 0; k < 64 and cur < 128; k++)
            {
                adj[k].push_back(i * 5 + j);
                cur++;
            }
        }
    }

    for (int kk = 0; kk < 64; kk++)
    {
        int nn = int(adj[kk].size());
        int cnt = 0;
        for (int mask = 0; mask < (1 << nn); mask++)
        {
            for (int i = 0; i < nn; i++)
            {
                if ((mask & (1 << i)) != 0)
                {
                    int ii = adj[kk][i] / 5;
                    int jj = adj[kk][i] % 5;
                    state[ii][jj][kk] = 1;
                    ans[ii][jj][kk] = 1;
                }
            }

            char hexa[16] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
'A', 'B', 'C', 'D', 'E', 'F'};

            uint64_t b = 1600;
            uint64_t l = 512;
            uint64_t c = 1024;
            uint64_t r = 576;
            int rounds = 24;
            int i, j, k;
            uint64_t tempstate[5][5][64];

            uint64_t current_round = 0;
            uint64_t column_parity[5][64];

            while (current_round < rounds)
            {
                //theta operation
                for (i = 0; i < 5; ++i)
                {
                    for (k = 0; k < 64; ++k)
                    {
                        column_parity[i][k] = 0;
                        for (j = 0; j < 5; ++j)
                            column_parity[i][k] ^= state[i][j][k];
                    }
                }

                for (i = 0; i < 5; ++i)
                {
                    for (j = 0; j < 5; ++j)
                    {
                        for (k = 0; k < 64; ++k)
                        {
                            state[i][j][k] ^= column_parity[(i + 4) % 5][k] ^ column_parity[(i + 1) % 5][k];
                            tempstate[i][j][k] = state[i][j][k];
                        }
                    }
                }

                //pi operation
                for (i = 0; i < 5; ++i)
                    for (j = 0; j < 5; ++j)
                        for (k = 0; k < 64; ++k)
                            state[j][((2 * i) + (3 * j)) % 5][k] = tempstate[i][j][k];

                //chi operation
                for (i = 0; i < 5; ++i)
                    for (j = 0; j < 5; ++j)
                        for (k = 0; k < 64; ++k)
                            tempstate[i][j][k] = state[i][j][k];

                for (i = 0; i < 5; ++i)
                    for (j = 0; j < 5; ++j)
                        for (k = 0; k < 64; ++k)
                            state[i][j][k] = tempstate[i][j][k] ^ (~tempstate[i][(j + 1) % 5][k] & tempstate[i][(j + 2) % 5][k]);

                ++current_round;
            }

            bool ok = 1;

            for (i = 0; i < 5; i++)
                for (j = 0; j < 5; j++)
                {
                    int foo = i * 5 * 64 + j * 64 + kk;
                    if (foo >= 512)
                        break;
                    ok &= (final_state[i][j][kk] == state[i][j][kk]);
                }

            for (i = 0; i < 5; i++)
            {
                for (j = 0; j < 5; j++)
                {
                    state[i][j][kk] = 0;
                    if (!ok)
                        ans[i][j][kk] = 0;
                }
            }

            if (ok)
            {
                break;
            }
        }
    }

    string pass = "";
    for (int i = 0, cur = 0; cur < 128; i++)
    {
        for (int j = 0; j < 5 and cur < 128; j++)
        {
            for (int k = 0; k < 64 and cur < 128; k++)
            {
                pass += char('0' + ans[i][j][k]);
                cur++;
            }
        }
    }

    cout << "The password is ";
    for (int i = 0; i < pass.size(); i += 8)
    {
        int cur = 0;
        for (int j = i; j - i < 8; j++)
        {
            cur = (cur << 1) + (pass[j] - '0');
        }
        if (cur)
            cout << char(cur);
    }
    cout << endl;

    return 0;
}
```

Assignment 7

● UNGRADED

GROUP

A5 - SURYADEVARA SAI KRISHNA
AJAY PRAJAPATI
A11 - GARIMELLA MOHAN RAGHU

View or edit group

TOTAL POINTS

- / 80 pts

QUESTION 1 Teamname	0 pts
QUESTION 2 Commands	5 pts
QUESTION 3 Analysis	50 pts
QUESTION 4 Password	25 pts
QUESTION 5 Codes	0 pts