Q1 Team Name
0 Points

Group Name

team_9

Q2 Commands
5 Points

List all the commands in sequence used from the start screen of this level to the end of the level. (Use -> to separate the commands)

go ->dive -> back ->back ->go->wave->back->back->"thrnxxtzy"->read->"the_magic_of_wand"->c
(Free the spirit by waving the magic wand at level 3 then enter in the level 4)
read -> password -> c -> plaintext (gives corresponding ciphertext on the screen)

Q3 Cryptosystem
10 Points

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

Data Encryption Standard (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. (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. At this level we got the hint when we type read after freeing the spirit at level 3. In the hint it was given that it can not be the 10 round DES and also 4 round DES is easy to be broken with 2 round characteristics. After entering the password we got $gqkqfgsitqljrtoknuoumqstjqtimkki$.
From the hint $twolettersforonebyte$ we got that each is represented using 4 bits and the inference of it is mentioned at later part of analysis. So, we started the Differential Cryptanalysis of 6 round DES. We have to use chosen plaintext attack to break DES as taught during the lectures.

2. We have also observed that for each plaintext we get a 16 char cipher output when generate the plaintext and ciphertext pairs. So, after analysis and from the hint we came to a conclusion that each char was of 4 bit so 16 such characters makes it 64 bit output. In the cipher text few characters of english alphabet never appears i.e a to e and v to z . So character mapping was clear that f to u was used which we could observe also. The mapping of characters f to u were represented by 4 bits where $'f'$ corresponds to $'0000'$and $'u'$ corresponds to $'1111'$ in chronological order. A pair of two letters forms a byte in the input-output blocks of 8 byte each.

3. We have used two 3-round characteristics with 0.0625 probability each. We could have used 4-round characteristic also but the overall probability reduced to 0.000381 in 4-round characteristics. The 3-round characteristics are
4008000004000000 and 0020000800000400.
To ensure this we need to do reverse of initial permutation which should be the XORs of a given input pair. We did this using "init_inv_perm.cpp" and we get
0000801000004000 and 0000080100100000.

4. We more than 1000 input output pairs(1000 * 0.0625=62.5) to get a frequency analysis of the keys at each round. We have generated 5000 such plaintext pairs using " Plain_gen.py" and then execute their xor to find the perfect input plaintext pairs which satisfies the characteristic equation. Two such files were generated for viz "Plain_texts1.txt" and " Plain_texts2.txt " for each of the characteristic equations.

5. Then we have generated corresponding cipher texts from the game by using "Cipher_gen.py " script and stored it in "Cipher_texts1.txt" and "Cipher_texts2.txt".

6. We have carried out $DifferentialCryptanalysis$ to get hold of the possible key with choosen plaintext attack. We did this by executing "Differential.ipynb" file. The Cipher texts from each file were read and converted each letter into binary with the mapping of f=0000, g=0001 and so on till u=1111. We have appliede inverse final permutation as per the fixed table and got the corresponding values of $L6R6$ and $L'6R'6$. We know that in order to get K6 (round key of 6th round) we need XORs of output pairs R5(right half at 5th round)=L6(left half of the round i.e the output). Then these xors were expanded using the fix expansion table. We need the output xor of S Boxes at last round. By doing the L5xorL'5 (=R4xorR'4) xor with R6xorR'6 we get the output xor of Permutation function at the last round. By doing inverse of permutation table we the output xor of S-boxes . we generate beta value pairs by checking the given alphaxors(output of Expansion) and for which S[beta1] xor S[beta2]=gamma xor values(Output of S-boxes).
We check the freq of keys in this calculation which comes out to be $45, 59, 37, 7, 37, 18, 14, 61$.

7. This freq of keys in decimal then converted to binary to get a possible guessed key where output of S3 were not fixed so we put 'X' in place of binary digits and we get $101101111011XXXXXX000111100101010010001110111101$. We did the permutation of the key by the fixed table and got $X11XX1XX01011X100XX11X11101X0110101X01001001X10X1111X001$This is 48 bit long where we don't know 6 bits. Hence we do a brute force method to get the rest of the 14 bits out of total 56 bits key. that comes out to be $01101110010111100111101110000001000011000111011111111001$. Out of this all the round keys were generated.

8. We had got a 32 character cipher text in the game when we typed "password" i.e $gqkqfgsitqljrtoknuoumqstjqtimkki$. In order to get the actual password for the level 4, firstly we convert the 32 char cipher text to a string of decimal numbers from reverse mapping of english alphabets f to u to corresponding binary and each (8 bits) gp of two characters were converted to corresponding decimal numbers and we get $27, 91, 1, 211, 235, 100, 206, 149, 143, 159, 123, 222, 75, 227, 117, 83$.

9. We make it into two halfs of 8 decimal numbers alongwith possible key value to get the password for the level 4 i.e "skwvgxdsgr000000". After removing the padded zeroes we get the password as $skwvgxdsgr$

Q5 Password
5 Points

What was the password used to clear this level?

skwvgxdsgr

Q6 Code
0 Points

Please add your code here. It is MANDATORY.

Assignment 4.zip Download

```
Large file hidden. You can download it using the button above.
```

init_inv_perm.cpp Download

```cpp
#include <iostream>

using namespace std;

int main() {
    char a[64],b[64]; int p[64];
    for (int i=0;i<64;i++) cin>>a[i];
    for (int i=0;i<64;i++) cin>>p[i];
    for (int i=0;i<64;i++) b[p[i]-1]=a[i];
    for (int i=0;i<64;i++) cout<<b[i];
    return 0;
}
```

Assignment 4

Graded