Q1 Team Name
0 Points

DECODERS

Q2 Commands
10 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)

enter->jump->jump->back->pull->back->back->enter->wave->back->back->thrnxxtzy->read->134721542097659029845273957->c->read->password->c->nmqpjacgiq->c

Q3 CryptoSystem
5 Points

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

6 ROUND DES(DATA ENCRYPTION STANDARD)

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.)

As the spirit told, we thought that the cryptosystem used here is 6 round DES and started cryptanalysis. We know that to break r round DES, we need r-2 round characteristics. Hence , we use 4 round characteristics in our analysis. We performed the following steps in cryptanalysis:

1. We know that the input XOR in 4 round characteristics is 0x405C0000 04000000 but we go through initial permutation before going to first round. Hence, we give 0x00009010 10005000 as the input XOR, which generates the above XOR after the initial permutation. We have generated 100,000 pairs of inputs which give the above XOR.

Program used:
Generating_input.ipynb (output: plaintexts.txt)

2. Now, we need to generate corresponding ciphertext values for the above plaintext values.
To do this, we have used Generating_scriptfile.ipynb which takes the plaintext values and generates a shell script file as the output (script.sh). This output is given as input to a command.sh file which gives these plaintexts as input to the ssh server and produces the output ciphertext (output.txt). After processing this file using Generating_ciphertext_file.ipynb, we get ciphertexts.txt which contains all the ciphertexts.

Program used:
Generating_scriptfile.ipynb (input: plaintexts.txt, output: script.sh)
command.sh (input: script.sh, output: output.txt)
Generating_ciphertext_file.ipynb (input: output.txt, output: ciphertexts.txt)

3. The ciphertext contains the letters as the values which need to be converted into binary format. From the hint we considered each letter as a 4 bit value. We noticed that all the letters are in the range 'd' to 's', which means the mapping from letters to 4 bit values is as follows:
d=0000
e=0001
f=0010

...
s=1111

Program used:
Ciphertext_to_binary.ipynb (input: ciphertexts.txt, outputs: binary_cipher.txt and bin_cipher.txt)

4. Now we have generated the ciphertexts in binary format on which we will apply the final inverse permutation, which will give us the left and the right half outputs of the 6th round i.e. L6,R6. We know that R5=L6 which will go through the expansion box of round 6 and get XORed with the key to generate the input to the S-box.
By using 4 round characteristic, we know that XOR at L5 is 0x0400 0000 with probability 0.00038 which get XORed with R6 pair and goes through inverse permutation to obtain the XOR of outputs of S-box.
The output pair of expansion box is given as a input to the S-box, since the key gets eliminated. Thus, we have XOR of the values as follows:
(i) Pair of expansion box outputs (ebox_op.txt)
(ii) XOR of pair of S-box inputs (sbox_ip.txt)
(iii) XOR of pair of S-box outputs (sbox_op.txt)

Program used:
Differential_cryptanalysis.ipynb (input: bin_cipher.txt, output: ebox_op.txt, sbox_ip.txt, sbox_op.txt)

5. Using the XOR of values of above three things, we can estimate the key values based on the characteristics. For each S-box, we will give all the XOR input values (sbox_ip.txt) and for each XOR, we will find the input pairs which give this XOR value. Now we will check for each input pair which ones give the XOR of the output the same as the ones that we have calculated (sbox_op.txt). From these input pairs, we will take the first element of the pair (can take second as well) from all pairs. Each of these first elements will be XORed with the output of expansion box to give us the possible value of the keys and we will store the occurrence of these keys. After doing this for all the pairs, each of the possible keys from 000000 to 111111 we will have certain occurrence. The

key with the maximum occurrence is selected as the final key for that particular S-box. We have taken the keys for which the difference between the average occurrence and the maximum occurrence is significant. From this, we have got the keys for the following S-boxes.

Possible Key for S-box 0 is 101101

Possible Key for S-box 1 is 110011

Possible Key for S-box 4 is 111100

Possible Key for S-box 5 is 101011

Possible Key for S-box 6 is 10011

Possible Key for S-box 7 is 101011

Program Used:
sbox_key_generation.ipynb

6. Now we have got the 36 bits out of the 56 bits key. We have applied the key transformations PC-2 and right shifts and generated the 56 bits key stream as follows:

XX1XX1XXX10X1X10XXX11XX10X0X0111111X00111111X01X0010X001
Here, X represents the missing values. We have 20 such missing values. We have generated all the possible key-values by replacing the unknown X by 0 and 1 (possible_keys.txt).

Program used:
Possible_keys.ipynb (output: possible_keys.txt)

7. In this step, we have performed the BRUTE-FORCE to obtain the final key by using the existing DES implementation (bruteforcekeys.ipynb). The final 56 bit key we got is as follows:
01101110010111100111101100000111111100111111101100100001
We have used this key to decrypt the password. After giving "password" as the input to the screen we get "rsqonrdgspfqikrghpsrnlshoolpgmmj". This string contains 32

characters, but we need to give 16 characters as the input. Hence, we have broken this string into 2 parts of 16 characters each. Both the parts are decrypted and then given as a password but it is not accepted. We then converted the above string into binary by replacing each character by 4 bit value (according to the mapping given in step 3) and taking the decimal value of each 8 bits to get 16 values.This gave us 239 219 174 3 252 45 87 227 76 254 168 244 187 140 57 150.
We have taken the first 8 and last 8 values as inputs and got the decrypted values as follows:

239 219 174 3 252 45 87 227 on decryption gives: 110 109 113 112 106 97 99 103

76 254 168 244 187 140 57 150 on decryption gives: 105 113 48 48 48 48 48 48

After combining these two values we got:

110 109 113 112 106 97 99 103 105 113 48 48 48 48 48 48

We noticed that all the above values are the ASCII range. Hence, we computed the ASCII character for each of the ASCII value and got "nmqpjacgiq000000" after removing the trailing 0's we got "nmqpjacgiq" which is accepted as the password and thus, we cleared this level.
Password: nmqpjacgiq

Programs used:
Extracting_Key.ipynb (input: possible_keys.txt, gives the final key)
DES.cpp (gives the ascii values of the decrypted password)

No files uploaded

Q5 Password

5 Points

What was the password used to clear this level?

nmqpjacgiq

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.

DECODERS_ASSIGNMENT4.zip Download

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

Assignment 4

GRADED

GROUP

Manthan Kojage
Akash Gajanan Panzade
Abhishek Dnyaneshwar Revskar
View or edit group

TOTAL POINTS

45 / 100 pts

QUESTION 1

Team Name **0** / 0 pts

QUESTION 2

Commands **10** / 10 pts

QUESTION 3

CryptoSystem **5** / 5 pts

QUESTION 4

Analysis **80** / 80 pts

QUESTION 5

Password **5** / 5 pts

QUESTION 6

Codes **-55** / 0 pts