

Q1 Team Name

0 Points

Group Name

team_ethereum

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->jump->jump->back->pull->back->back->enter->wave->back->back->thrnxtzy->read->the_magic_of_wand->read->password->qqrjkecmhd

Q3 Cryptosystem

10 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. (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.)

EXPLANATION:

1. We used **read** on the first screen because there was a panel nearby, but nothing was written there. Therefore, we decided to use **enter**. We used **jump** because we were at the edge of the lake on the following screen. We again came out. So, we used **jump** again

A magic wand was seen there and tried **pull**. However, we went out of breath and died. As a result, we entered the aforementioned commands once more, switching **pull** for **back** and then used dive again. We used **pull** now and obtained the wand.

2. Using a series of backs, we returned to the first screen at this point. And attempted **read** command, however it was still empty. We eventually realised that it has something to do with the chapter's name, THE SPIRIT, and we realized that level 3 contained a spirit.

3. We then returned to level 3 and used **enter** command to get to the second screen, where we used wave to free the spirit. After that, we clear that level using the same commands as in chapter 3 (thrnxtzy -> **read** -> the_magic_of_wand). We then received the question **read** by spirit from the first screen of the 4th chapter on using **read** command.

We got the following ciphertext on entering the “password” on level 4 screen which is as mentioned:

“homfpkmjntpgtiihilgoghgplmksgnsi”

4. We deduced from the spirit's hint that this level's cryptosystem is either 4 round DES or 6 round DES. There was a very small chance that it would be a 10-round DES. So, we began by assuming a 6-round DES. We deduced from the hint

“twolettersforonebyte” that each letter is represented using 4 bits, so only 16 out of 26 letters are possible here. We discovered that letters from f to u were present in ciphertext after providing multiple random plaintexts as input, so when generating plaintext for the attack, we only used letters from [f,u]. Because DES blocks are 8 bytes in size, each block contains 16 letters. Thus, we proceeded by mapping letters f-u to 0-15 as follows:

f : 0000, g : 0001, h : 0010, i : 0011, j : 0100, k : 0101, l :

5. To break DES encryption, a chosen plain text attack is used. We used **differential cryptanalysis** to generate plain text pairs, which we then passed to the system to generate corresponding ciphertext pairs, which we then used to find the key and decrypt the above ciphertext.

STEPS OF ANALYSIS

1. We produced 5000 pairs of plain text for each characteristic using **generator.py**. We used two three-round characteristics with a 0.0625 probability each. The characteristics are "00 20 00 08 00 00 04 00"

2. In order to create 5000 pairs that satisfy the characteristic "40 08 00 00 04 00 00 00," we made sure that their xor was "00 00 80 10 00

00 40 00," which was obtained by applying inverse initial permutation to the aforementioned characteristic.

3. In a similar manner, we created 5000 pairs of plaintext that satisfied the "00 20 00 08 00 00 04 00" characteristic. We made sure that these pair's xors were "00 00 08 01 00 10 00 00," which was obtained by applying inverse initial permutation to the previously mentioned characteristic. **plain1.txt** and **plain2.txt** are the respective files where these plain texts are stored respectively.

4. To create ciphertexts that matched the plaintexts, we ran **script1.py** and **script2.py** and thereby saved the results in **cipher1.txt** and **cipher2.txt**.

5.

To find the secret key, differential cryptanalysis

DES_6th_Round_Analysis.py is used to complete the steps given below.

- We start by reading cipher1.txt. For each ciphertext, we turn each letter into binary using the mapping where f is 0000 and u is 1111.
- We applied the inverse final permutation in order to get **(L6, R6)** and **(L'6, R'6)**.
- We know that $R5 = L6$, so we use $R5$ and $R'5$ to find the output of the expansion box and the input XOR of sboxes for the 6th round.
- The ciphertext pairs have been XORed according to the differential. Followed by expanding the Right side block of Round 5. The expanded output will then get Xored to compute the S-Box input. $L5 \oplus R6$ was then computed and Xored the output of the S-Box.
- $L5 = 04\ 00\ 00\ 00$ for first characteristic and $L5 = 00\ 00\ 04\ 00$ for second characteristic. Then we performed $L5 \text{ XOR } (R6 \text{ XOR } R'6)$ then applied inverse permutation to get output XOR of sboxes for 6th round.

Let,

$$E(R5) = \alpha_1 \alpha_2 \dots \alpha_8 \text{ and } E(R5') = \alpha_1' \alpha_2' \dots \alpha_8'$$

where,

$$|\alpha_i| = 6 = |\alpha_i'|$$

and,

$$k_6 = k_{6,1} k_{6,2} \dots k_{6,8}$$

and,

$$\beta_i = \alpha_i \oplus k_{6,i} \text{ and } \beta_i' = \alpha_i' \oplus k_{6,i}$$

We know that,

$$\alpha_i, \alpha_i', \beta_i \oplus \beta_i' \text{ and } \gamma_i \oplus \gamma_i'$$

A $8 * 64$ key matrix has been created to store the number of times a key $k \in [1, 64]$ had satisfied the possibility of being a key to S_i box, where $i \in [1, 8]$.

We find the set

$$X_i = \{(\beta, \beta') \mid \beta \oplus \beta' = \beta_i \oplus \beta_i' \text{ and } S_i(\beta) \oplus S_i(\beta') = \gamma\}$$

Then for each $k \in [1, 64]$, we determine whether

$$\alpha_i \oplus k = \beta \text{ and } (\beta, \beta') \in X_i \text{ for some } \beta'$$

On satisfying the above condition for S_i box, the key[i][k] will be incremented by 1.

6. Analysis of the aforementioned characteristic 40 08 00 00 04 00 00 00 is that we get partial key using S2, S5, S6, S7, S8 as 51, 43, 49, 25, 56 as input to these sboxes is 0 in round 4. The above procedure has been repeated similarly for ciphertexts in ciphert2.txt.

The above analysis gave the following results:

S-box	Max	Mean Key	
S1	332.0	170	61
S2	797.0	196	51
S3	302.0	165	37
S4	246.0	164	7
S5	403.0	171	43
S6	786.0	192	49
S7	515.0	178	25
S8	504.0	176	56

Analysis of the aforementioned characteristic 00 20 00 08 00 00 04 00 is that we get partial key using S1, S2, S4, S5, S6 as 61, 51, 7, 43, 49 as input to these sboxes is 0 in round 4.

S2, S5, S6 are common in these characteristics, and key bits deduced from both these characteristics are the same for before mentioned sboxes. 42 out of 56 bits of the key have been found successfully.

The above analysis gave the following results:

S-box	Max	Mean Key	
S1	332.0	170	61

S2	797.0	196	51
S3	302.0	165	37
S4	246.0	164	7
S5	403.0	171	43
S6	786.0	192	49
S7	515.0	178	25
S8	504.0	176	56

48 bit Key for Sbox is

111101110011XXXXXX000111101011110001011001111000

7. As input to S3 was never zero 'X' are inserted in the position of S3. This will be then converted into a 56-bit key and thereby applying Key schedule PC2, we obtain

X11XX1XX01011X100XX11X11001X1111001X10011000X00X111
1X001

8. We used the brute force method to find the missing bits, which involved iterating through all 2¹⁴ possible keys. The system received "fghijklmnopqrstu" as input plaintext. The cipher is "kfnrhmrhghjqghf". Then, for each possible key, we encrypted the plaintext with this key to see if we got the cipher described above. The actual key is the key that matches the output of encryption and the above cipher. The actual 56 bit key is

0110111001011110011110110010111100111001110

On obtaining the 56 bit key, we determined the 48 bit round key for each round.

The round key in binary for each round are as mentioned below:

Round 0 key is

11101100010011111000011100101111100111110001110

Round 1 key is

011011110011011101100010000011100011011110101010

Round 2 key is

11101010110111001110110111111000111100101100101

Round 3 key is

11011001111000110101101001100010110010101111010

Round 4 key is

001001001101111110111011110101011011110100011011

Round 5 key is

111101110011100101000111101011110001011001111000

9. Password Decryption: -

"homfpkmjntpgtiihilgoghplmksgns" was first converted into binary and then into decimal. It was then divided into two parts as at a time

DES only works on 8 bytes of plaintext, to get {41,112,165,116,142,161,227,50} and {54, 25, 18, 26, 103, 93, 24,211} where each block is 8 bytes. This is then passed one at time in des file.

We received **“qqrjkecmhd000000”** after decryption.

Now, we thought "000000" was padding, we tried the password **”qqrjkecmhd”**, and we succeeded in completing the level.

Q5 Password 5 Points

What was the password used to clear this level?

qqrjkecmhd


Q6 Code 0 Points

Please add your code here. It is MANDATORY.

▼ team_ethereum_code.zip

 [Download](#)

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

Group
ALLAN ROBEY
DIVYESH DEVANGKUMAR TRIPATHI
AVNISH TRIPATHI
 View or edit group

Total Points	
100 / 100 pts	
Question 1	
Team Name	0 / 0 pts
Question 2	
Commands	5 / 5 pts
Question 3	
Cryptosystem	10 / 10 pts
Question 4	
Analysis	80 / 80 pts
Question 5	
Password	5 / 5 pts
Question 6	
Code	0 / 0 pts