Q1 Teamname
0 Points

cryptoEngineers

Q2 Commands
10 Points

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

1. go/enter (It takes us to the edge of the lake in the right side)
2. dive/jump (to dive into the lake)
3. dive (to dive into lake again)
4. back (to come at surface and dive back after inhaling deeply)
5. pull (to pick up the magic wand)
6. back (It takes to start of level 4)
7. back (It takes us to the start of level 3 i.e. main chamber)
8. enter/go (It takes us to small chamber)
9. wave (It frees the spirit)
10. back (returned to small chamber)
11. back (returned to main chamber)
12. thrnxxtzy
13. read (to read the glass panel)
14. 3608528850368400786036725
c to continue
15. read (It takes us to the magical screen)
16. password (It provides ciphertext)

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. (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.)

After reading to magical screen, and following the word whispered by the spirit, we got to know the following things:

1. The Code used is DES (not sure about the number of rounds).
2. Coded Password can be seen by whispering the word "password".
3. Two letters corresponds to one byte.

As per the spirit, 1 character maps to 4 bits, so, there can be atmost 16 characters. After generating few encoded texts, we observed that the characters in the encoded texts were between 'f' and 'u', i.e. total 16 distinct characters.
Hence, we assumed that 'f' maps to 0b0000, 'g' maps to 0b0001 . . . . . 't' maps to 0b1110 and 'u' maps to 0b1111.
As spirit was confused in number of rounds(4,6 or 10) of DES, we thought to start with 6-round DES as it was moderately difficult in compare to 4-round and 10-round DES.

Differential Cryptanalysis for 6-Round DES

Referring to the resources (DES-differential-cryptanalysis.pdf) provided for this assignment, we got to know that 6-round DES can be decrypted using two 3-round characteristics and these characteristics are,

$$0\text{x } 40080000\ 04000000$$
$$0\text{x } 00200008\ 00000400$$

(1) **Taking first characteristics 0x 40080000 04000000**.

Now, we need input pairs such that their XOR's after Initial Permutation(IP) equals to $0\text{x } 40080000\ 04000000$, hence, we generated 1000 pairs (2000 inputs) using code $generatePlaintext.py$ as these 1000 pairs were enough to find most frequent keys. Also, we generated ciphertexts corresponding to input plaintexts using $script.sh$

3 round characteristics :

$$(0\text{x}40080\bar{0}\ 04000\bar{0}, \tfrac{1}{4}, 0\text{x}04000\bar{0}\ \ \bar{0}\bar{0}, 1, 0\text{x}\bar{0}\bar{0}\ 04000\bar{0}, \tfrac{1}{4}, 0\text{x}04000\bar{0}\ 40080\bar{0})$$

where $\bar{0}$ stands for 16 bit string $0\text{x}0000$.

After round 3, we get XOR value $0\text{x } 04000000\ 40080000$, with probability $p = \frac{1}{16}$. Now, for round 4, input XOR's of S-boxes $S2, S5, S6, S7, S8$ are zero and hence, output XOR's to corresponding S-boxes also zero with probability, $p = \frac{1}{16}$. After permutation of output XOR's of S-boxes and XORing with left half ($0\text{x } 04000000$), we get right half of 4th round output.

This output right half of 4th round will be the output left half of 5th round with probability, $p = \frac{1}{16}$, which is then XORed with right half of XOR of inverse final permuted ciphertext pair. (Since, we are targetting XOR bits corresponding to S-box $S2, S5, S6, S7, S8$ and they are zero. Also, XORing it with right half of XOR of inverse final permuted ciphertext pair, it will not affect the XOR of ciphertext pair. Hence this XORing can be omitted). This result is then reverse permuted and we get output XOR values of corresponding S-boxes of 6th round.

Due to the output XOR values (i.e. zero) for required S-boxes($S2, S5, S6, S7, S8$) in round 3, we get the output XOR values of S-box $S2, S5, S6, S7, S8$ of round 6.

Left half of XOR of inverse final permuted ciphertext pair will be the right half of input XOR to the 6th round, which is then, expanded and we get input XOR values to 6th round S-boxes.

Now, we have both output XOR values and input XOR values of S-boxes $S2, S5, S6, S7, S8$ of 6th round with probability, $p = \frac{1}{16}$.

For each of the S-box $S2, S5, S6, S7, S8$, we find 4-bit output pairs such that their XOR is equal to output XOR value of that corresponding S-box.
Now, for that output pair, we generated corresponding input pair of that S-box using S-box table and checked whether XOR of generated input pair is equal to input XOR value of that S-box or not. If it is equal, then, that inputs can be the possible inputs of that S-box.
Also, we have right half of 6th round input which is nothing but the left half of inverse final permuted ciphertext. This is, then, expanded and XORed with possible inputs, to get possible 6-bit key ($k_{6,i}$) of that i^{th} S-box.

Now, the most frequent 6-bit key ($k_{6,i}$) will be the final 6-bit key corresponding to that S-boxes ($S2, S5, S6, S7, S8$) i.e. we get 30 bits out of 48 bits of subkey and these are

XXXXXX 111011 XXXXXX XXXXXX 001010 011000 010001 11111

where X's are unknown bits.

(2) **Taking characteristics 0x 00200008 00000400**.

3 round characteristics :

$$(0\text{x}00200008\ \bar{0}0400, \tfrac{1}{4}, 0\text{x}\bar{0}0400\ \ \bar{0}\bar{0}, 1, 0\text{x}\bar{0}\bar{0}\ \bar{0}0400, \tfrac{1}{4}, 0\text{x}\bar{0}0400\ 00200008)$$

where $\bar{0}$ stands for 16 bit string $0\text{x}0000$.

For this characteristics, after 3rd round, we get XOR values = $0\text{x}\ 00000400\ 00200008$.
For round 4, in this case, input XOR values of S-boxes $S1, S2, S4, S5, S6$ are zero and hence, corresponding output XOR values of that S-boxes are also zero with probability, $p = \frac{1}{16}$.

Similar to first characteristics, here also, we will perform same steps from generating plaintext pairs and ciphertext pairs to get 30 bits out of 48 bits subkey corresponding to these S-boxes $S1, S2, S4, S5, S6$ as

111101 111011 XXXXXX 000111 001010 011000 XXXXXX XXXXX

where X's are unknown bits.

We get key bits corresponding to $S2, S5, S6$ S-box from 30 bits of partial subkeys of both characteristics equal. This ensures the correctness of our 30 bits of partial subkeys. Combining partial subkeys of both characteristics, we get partial 48 bit subkey of round 6 as
111101 111011 XXXXXX 000111 001010 011000 010001 111111
where X's are unknown bits.

After applying reverse key scheduling on partial 48 bit subkey of 6th round, we get partial 56 bit key as
X11XX1X X01011X 100XX11 X11101X 1011101 X001011 00X10X0
where X's are unknown bits.

This partial 56 bits key contain 14 unknown bits. Taking all possible values of unknown bits (0 or 1), we generated 2^{14} possible 56 bit keys.
Now, these 2^{14} possible keys were brute forced to get 56 bit final key. Further, this 56 bit key is converted to 64 bit key by inverse permuted using PC1 and adding parity bits.

Encrypted Password : mfsrnosmfgfsthnqitnsrgiulutghupu
56 bit final key : 011011100101111001111011101010111011001011000

64 bit final key : 00111011111000111100110111100110011010111011

After decryption, we get 128 bits in which each byte corresponds to a ASCII character. After converting each byte into its ASCII character, we get **rivpfihqiw000000** and after removing trailing zeroes, we get password **rivpfihqiw**.

Programs :

generatePlaintext.py : It generates plaintext pairs whose XOR after Initial Permutation (IP) is $0x\ 40080000\ 04000000$ and $0x\ 00200008\ 00000400$ and creates file

inputPlaintexts.txt.
generateCiphertext.py : To generate ciphertexts corresponding to input plain texts.
script.sh : It first connects to server, then generates cipher texts corresponding to input plain text pairs using a program *generateCiphertext.py* which takes *inputPlaintexts.txt* as input.
differentialCryptanalysisch1.py : It generates 6 bits of subkey corresponding to S-box $S2, S5, S6, S7, S8$ using first characteristics, i.e. 30 bits.
differentialCryptanalysisch2.py : It generates 6 bits of subkey corresponding to S-box $S1, S2, S4, S5, S6$ using second characteristics, i.e. 30 bits.
keyGeneration.py : It creates partial 56 bits key from partial 48 bits key and generates all possible 56 bit keys in *possibleKeys.txt*.
DES.py : It brute forced every possible 56 bit key from file *possibleKeys.txt* and find correct 56 bit key and also provide us the final correct 64 bit key.
decryptionDES.py : The final correct 56 bit key is used to decrypt the coded password in this program and also prints it.

Note : All above programs were according to the progress of level 4 (i.e. on read command, reaching directly to magical screen). If progress get reset, then, lines of some programs have to be uncomment (it is mentioned in program what to change) to run properly.

References :

1. DES-differential-cryptanalysis.pdf provided in Resource section.

No files uploaded

Q5 Password

5 Points

What was the final command used to clear this level?

rivpfihqiw