CS641 - Assignment 4

Nishtha - 180489

Text on the wall :

“This is a magical screen. You can whisper something close to the screen and the corresponding coded text would appear on it after a while. So go ahead and try to break the code! The code used for this is a 4-round DES, so it should be easy for you!! Er wait ... maybe it is a 6-round DES ... sorry, my memory has blurred after so many years. But I am sure you can break even 6-round DES easily. A 10-round DES is a different matter, but this one surely is not 10-round ...(long pause) ... at least that is what I remember. One thing that I surely remember is that you can see the coded password by whispering ’password’. There was something funny about how the text appears, two letters for one byte or something like that. I do not recall more than that. I am sure you can figure it out though ...”

Cryptanalysis

- **Commands to get to cipher**

  Go -> dive -> dive-> back-> pull-> back-> back-> go-> wave-> back-> thrnxxtzy-> read-> 360852885036840078603672 5 -> c -> read -> password

- **Analysis**

  We got the encrypted text by typing "*password*" and the text was "*krsunjhoqfhpurpkksomjjoulmnrsfuj*". We tried so many inputs and found out that the 16 letters in the output was from *f*−*u*. It was given that the DES can be of any number of rounds so we started with 6 round as 2, 4 round are bit easy and if 6 round fails then we'll go to 4 or 10 round DES. Also we tried that our input consists of only 16 letters from *f*−*u* and these were mapped to number 0−15. We tried differential iterative characteristic of 6 round DES 405*c*000004000000 to break the code which gives the differential 0054000004000000 after 4 rounds with probability 0.000381. We need approx one lakh pairs of plain and cipher text for this. Then we used *input*.*py* and stored the output in *input*.*txt* to generate 100000 random inputs and input pairs whose XOR is equal to 0000902010005000 which is inverse initial permutation of 405*c*000004000000.

  Then we used *output*.*sh* and run it on the game to generate output corresponding to inputs that we just generated and stored these in *output*.*txt*. The output has some

waste information and to get only the ciphertext, we used clean_out.py which produces correct output in clean_output.txt.

Next, we use the differential.py script where we firstly reverse the final permutation of R6 block and get the output of 6th round of DES. The left half of this output obtained is equal to the right half output of round 5. This is expanded to get the input differential of S-Boxes using ExpansionBox and contained in a1.txt. Next the right half of the output of 6th round is inversely permutated to obtain the output differential of th S-Box stored in a2.txt. At last, picking from every two entries x1, x2 of the output of 6th round of DES, we pick left half of x1 and store its expanded form in a3.txt
After that using the sbox.py script, we extract the 6 bit key corresponding to that S box with the help of a3.txt for each correct input output differential pair.