CS641A Assignment-4

Sherlocked

Bholeshwar Khurana (170214) **Yaghyesh Chouhan (170813)** **Sarthak Dubey (180674)**

1 How we reached the cipher text:

1. On the first screen, there was a door with a panel but there was nothing written on it, so we used **go**.
2. We reached the edge of a lake, where we used **dive**. We came out so we used **dive** again. We found a magic wand, and we tried **pull** but we ran out of breath and died.
3. So, we got to the magic wand again using aforementioned commands, and used **back** and then **dive** again. Now, we used **pull** and got the wand.
4. We went back to the first screen using **back**. We tried to read the panel again but it was still blank. We then figured out that the chapter's name is *THE SPIRIT* and there was a spirit in level 3.
5. We went back to level 3 using **back**, used the command **go** to reach the second screen where we used **wave** to free the spirit. Then we used the commands we found in level 3, i.e, **thrnxxtzy** then **read** and then the password **kgg_mnzcr_yv**.
6. We reached the first screen of level 4 again where we used **read** and found the question read by the spirit.

2 How we cracked the cipher text:

1. Knowing it is a 3-round DES, we used the differential cryptanalysis method taught in class for cryptanalysis of 3-rounds, which is a chosen plaintext attack where input pairs are chosen such that the differential of right side (32 least significant bits) is equal to 0.
2. We were given that 2 letters constituted one byte, hence each letter is of 4 bits. Hence total possible letters in input and output must be 16. Also the input and output size of DES is 64 bits, which is 8 bytes or 16 letters, and the output size was indeed 16 letters. We did a frequency analysis of output and found that it consisted only of letters between 'f' and 'u', and as we had 16 total letters, with 4 bits for each letter, *we mapped letters 'f' to 'u' to numbers from 0 to 15 respectively*.
3. First, we started by generating 10 pairs of random 64 bits, such that the last 32 bits of a pair are same, i.e, has xor value equal to 0. This was done in **1_random_inputs.py** and the pairs were stored in **random_inputs.txt**.
4. Using these random bits, we generated input for DES by first permuting the 64 bits using inverse initial permutation (IP^{-1}), and then mapped the 4-bit blocks to letters between 'f' and 'u'. This was done in **2_actual_inputs.py** and the inputs were stored in **actual_inputs.txt**.
5. Using the script **3_generate_outputs.py**, we got corresponding ciphertexts to the plaintexts we generated in previous step, and stored them in **generated_outputs.txt**.
6. After obtaining plaintext and corresponding ciphertexts, we converted the ciphertexts to binary and permuted them using initial permutation (IP) (which is inverse of final permutation) and then swapped their left and right parts. This step was carried out by **4_outputs_after_IP_swap.py** and the outputs were stored in **outputs_after_IP_swap.txt**.

7. We then calculate the xor of input and output pairs using **5_inputs_xor.py** and **6_outputs_xor.py**.

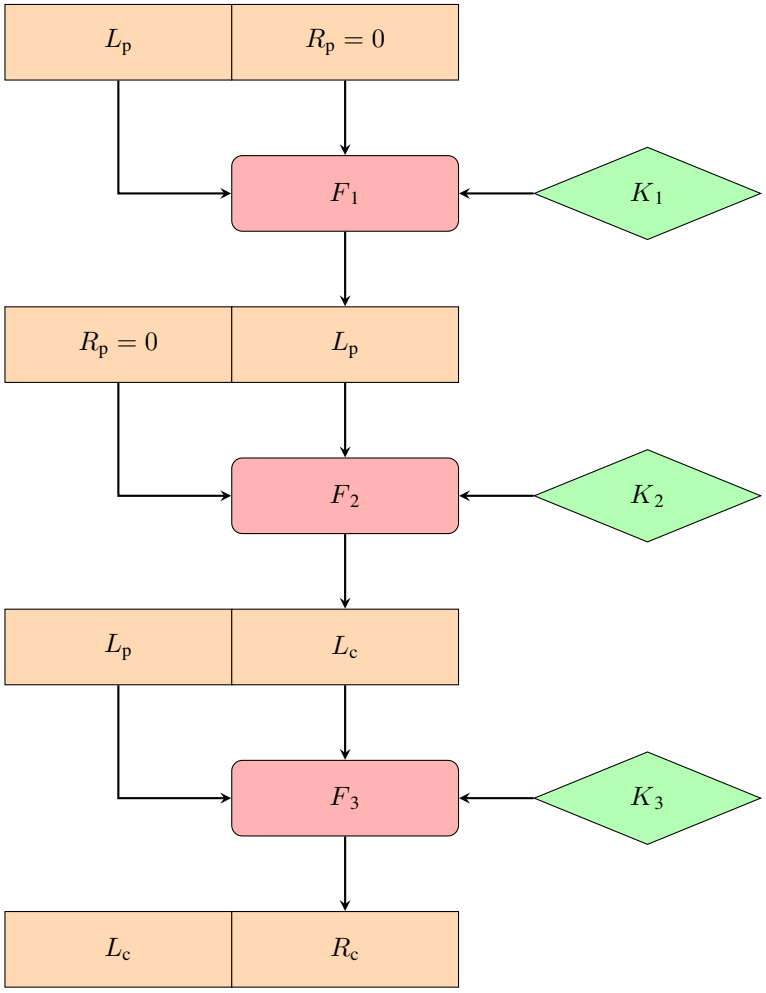

Fig. 1

8. Fig. 1 shows the values of differential at different stages of encryption, where R_c, L_c are ciphertexts and R_p, L_p are plaintexts, also $R_p = 0$ and all of the above values (xor as well as individual values of plaintext and ciphertext pairs) are known. We know the individual values of pairs constituting L_c, hence we know the values after expansion. We also know the input as well as output xor of each sbox in F3. Now, for every possible key combination of K3 for each of the 8 sboxes, we xor it with the two L_cs after expansion and pass the value obtained through sbox and calculate their xor. If the xor value mathes with the one obtained using R_c, we consider the key value a possible one. After repeating this for all 10 input-output pairs, we are left with only one possible K3. Hence, K3 is obtained to be

$$K3 = 011101110000001000011001000110100011100110011101$$

This is done using **7_k3_breaker.py**. One can directly obtain K3 by running the sript **k3_break.sh**.

9. We use the 48 bit K3 obtained to get the 56 bit DES-key using **reverse_key_scheduling.py** which reverses the key scheduling algorithm of DES. Note that 8 bits of 56 bit DES-key are still unknown.

10. We find the remaining 8 bits of DES-key using bruteforce using **key_brute_force.py** and **des_3round.py** (which is the implementation of 3 round des using given constants.py). We check all the 2^8 possibilities and check whether they produce the correct outputs for the taken inputs. We thus obtain the 56 bits key:

56-bit key = *10110101010110000100000010001010111011100011100110100*

11. We now have the 56-bit key. We obtain round-keys, i.e. $k1, k2, k3$ using the 56 bit key using key-scheduling algorithm of DES. Once we have the round keys, we decrypt the given ciphertext(password) by using $k3, k2, k1$ respectively in DES and passing the encrypted text as the input. Hence, we decrypt the password *hshjsijnrkptugsififnluhuphpkpqnm* using **inverse_des.py** and obtain the password:

tkjkkktmpghsqfosqkjsjihgsnsnsrij