CS641A
Assignment-4

Team Name: noname
M Anirudh(13377)
Sivasankar C Nair(13702)
Khemkaran Sevta(13349)

Part-1

The first part of the assignment involved solving riddles by entering certain commands using clues from the given text. go -> dive -> dive -> back -> pull -> back -> back -> go -> wave ->back -> thrnxxtzy -> read -> zun_rksm_ul_mlgi -> c -> read -> password .After this we get the encrypted password which was encrypted using DES .We could enter any plaintext and get cipher text .

Part-2

The second part involved decrypting a password which was encrypted using either 4, 6 or 10 round DES. 4 round DES is very easy to break so we started with 6 round. It was given that two letters were for one byte and DES has a block size of 64 bits or 8 bytes so 16 letters represented 1 block size. Because there were only 16 letters in the output which we found after doing frequency analysis which were from 'f' to 'u' so we thought that input also consists of these letters only as only 16 letters could be represented by four bits and each alphabet was mapped with a number from 0 to 15. The differential iterative characteristic of 6-round DES '**405c0000 04000000**' was used to break the code which gives the differential **'00540000 04000000'** after 4 rounds with probability 0.000381 for which we need approximately one lakh pairs of plaintext-cipher text pairs. We used two cpp programs **rand.cpp** and **random_input.cpp** where **rand.cpp** is for generating 1 lakh random strings and stored in **random.txt** and **random_input.cpp** is used to get input pairs whose xor is equal to '**0000902010005000**' which we get from inverse initial permutation of '**405c0000 04000000**' and stored in **input_random.txt**.

Now, we ran a python script (**script.py**) using **input_random.txt** to get corresponding outputs for these 1 lakh pairs of input. The output was stored in **output.txt** and the waste outputs were removed by **clean.cpp** and then stored in **output_clean.txt**.

Now we run the file **alphabettobinary.cpp** which takes **output_clean.txt** as input and gives outputs in **output1.txt** after converting alphabets into 4 bit binary code. Then differential.cpp is run in which output1.txt is taken and three different outputs are printed in **alphaxor.txt**, **betaxor.txt** and **reversefinalperm.txt**. In this program, the block is first reverse final permuted and stored in **reversefinalperm.txt**. Then the whole odd and even blocks(they both are pairs) are xor with each other. Alpha is value just after the xor with key 6,beta is value just after the sbox in 6^{th} round and gamma is just before the taking xor with key 6 and after expanding R5 which is also equal to L6.The left hand side bit is expanded and that is equal to gamma1 xor gamma2 which is also equal to alpha1 xor alpha2 which is stored in **alphaxor.txt**. The right hand is xor with xored L5(equal to R4 which is probabilistically equal to **'04000000'** and then inverse permuted to give beta1 xor beta2 stored in **betaxor.txt**.

The **gamma.cpp** computes the gamma1 values after expanding the left hand side of the reverse final permuted block from **reversefinalperm.txt** and stores it in **gamma.txt.**

Then we used **sbox.cpp** program to get the values of alpha1 and alpha2 corresponding to alpha1 xor alpha2 and whose S(alpha1) and S(alpha2) is also equal to beta1 xor beta2 .For each permuted 6 bits of alpha1 we calculated value of alpha2 by taking xor of alpha1 and (alpha1 xor alpha2). After this we compare the value of S[alpha1] xor S[alpha2] with beta xor if these alpha1 and alpha2 satisfy these condition than these are the possible values of alpha1 and alpha2.

Team Name: noname
M Anirudh(13377)
Sivasankar C Nair(13702)
Khemkaran Sevta(13349)

After this, We calculate key 6 by taking the xor of the candidate alpha1 with corresponding gamma1 and kept a counter for it. The key will repeat itself more frequently as compare to others. These frequencies are stored in **key.txt** file.

Key Block	1	2	3	4	5	6	7	8
Key Block Value	45 (101101)	51 (110011)	-	-	20 (010100)	60 (111100)	2 (000010)	46 (101110)
Max. Frequency	5463	5711	-	-	6034	5671	5671	5595
Ave. Frequency	4214	4174	-	-	4354	4251	4224	4209

Now, we get 6 blocks values out of 8 blocks but we could not get the key values from S-boxes S3 and S4. So now we have knowledge of 36 bits out of 56 bits.

The **keyscheduling.cpp** gives us the knowledge how the bits were permuted giving output in scheduledkey.cpp.

The key was manually mapped and found to be

XX1XX1XXX10X1X10XXX11XX10X0X0001110X00110001X01X0100X011

After this we apply brute force on the rest 20 bits of the key using **key_perm.cpp** and stored these possible keys in **key_out.txt**. Then we run it through **desbruteforce.cpp** and check whether a given input is encrypted in the same way or not. If it is, then that is the key.

Given input **: ffffffffffffffff {0,0,0,0,0,0,0,0}**
Given output : **tjtlgnslrqutjjpr {228,230,24,214,203,254,68,172}**
Key **: 01101110010111100111101100000001110100110001111101000011**

Then we run 2 times **decryptDES.cpp** and decrypt the password using the key.

Encrypted Password : **tjmituupspuqmiphrfuhjtgnksfoonpn {228,115,239,250,218,251,115,162,**
192,242,78,24,93,9,152,168}

Decrypted Password : **{110, 120 ,117 ,109 ,106 ,99, 104 ,104 ,**
107,119,48,48,48,48,48,48}

As the numbers lied between 97 and 122 we thought that these could be ASCII codes and the answer could be the conversion into characters. The 48 represented 0 and thus was just to pad.

Answer was found to be **'nxumjchhkw'.**

CS641A
Assignment-4

Team Name: noname
M Anirudh(13377)
Sivasankar C Nair(13702)
Khemkaran Sevta(13349)