Q1 Team Name

0 Points

Group Name

SecureSavants

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)

The commands used to solve this level are executed in the following sequence:

go->dive->dive->back->pull->go->back->enter->wave->back->back->thrnxxtzy->read->the_magic_of_wand->c->read->skwvjxdsgq

Q3 Cryptosystem
10 Points

What cryptosystem was used at this level? Please be precise.'

The cryptosystem used in this level is 6 round DES[Data Encryption Standard].

Actually the spirit told us that the cyptosystem used in this level is either 4 round DES or 6 round DES and it is highly unlikely that it is 10 round DES as per the spirit.So we proceded by checking for 6 round DES and indeed the cryptosystem came out to be 6 round DES as we were able to get through the level.

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.)

GETTING TO THE CIPHERTEXT

1. To reach the cipher text in this level we used the command used in question 2 in sequence.
2. Actually what happened is that we firstly entered using the enter command to get into the chamber.
3. Now we see a lake and then we dive into the lake using the **dive** command.After taking deep breath we again dive into the lake and get the magic wand with the help of **pull** command.
4. Now after using a series of backs we returned to the

first screen and attempted to read the command , however it was still empty.Then we realized that it has something to do with the level 3.
5. We then returned to level 3 and used the enter command to get into the second screen where we used the **wave** command to free the spirit.
6. After this we cleared that level using the same commands as level 3 which were (thrnxxtzy->read->the_magic_wand) .
7. After this the spirit talk to us and gave us hint that the cryptosystem used in this level is either 4 round DES or 6 round DES and it is highly unlikely that it is 10 round DES as per the spirit.
8. Also the spirit told us to enter the word password to get the encrypted password . We entered the command password and we got the encrypted password as **"fpljsgojpkhukunguknt ppkngjfsjgkm"**. Now our goal of this level is to decrypt this password.

Analysis

1. As per the hints of the spirit the spirit had told us that the cryptosystem used is either 4 round DES or 6 round DES it highly unlikely that the cryptosystem used is 10 round DES.
2. So we first started by assuming that the cryptosystem used in this level is 6 round DES. As it is highly unlikely that the cryptosystem used is 10 round DES as per the hint given by the spirit if 6 would have failed then we would have proceeded with 4 round DES.
3. Also the spirit told that here in this cryptosystem we are using **two letters for one byte**. This means for each letter we are using half byte to represent the letter which implies 4 bits.
4. But we know that 2^4 is 16 therefore our ciphertext contains only 16 letters and not 26 letters.

Checking Which Letters Are Being Used In Th

1. So to check what are the letters which are being used in the ciphertext we with the help of code.We first randomly generate plaintexts and store these all randomly generated plaintexts in **plaintext1_r.txt** file.
Some of our randomly generated plaintexts are:

$$irhnipoqrqfprlrs$$
$$irhnqpnqrqfpnlrs$$
$$hihfidqdeefpffrh$$
$$hihfqdpdeefpjfrh$$
$$khempdigdrrofiqn$$
$$khemhdhgdrrojiqn$$

2. After generating random plaintexts now we have employed the chosen plaintext attack technique to obtain the corresponding ciphertext of the randomly generated plaintext. For this we made use of pexcept library.
3. What actually we did is that with the help of python code present in the file **vikrant_final.py** we accessed the shell and logged in into the server game with the help of our credentials. We then executed the commands as given answer 2 in the exactly given sequence so as to reach the screen from where we can do the chosen plaintext attack.
4. After this we one by one picked up all the 10000s plaintexts and obtained their corresponding ciphertext with the our python script present in the file **vikrant_final.py** and store the corresponding ciphertexts in the file **ciphertext1_r.txt**.
5. Some of the obtained ciphertexts are:-

$$fqiqorrsimflipmt$$
$$nsfhnfojronhiijj$$
$$stmritftjttfsljg$$
$$tjhtmtiupqufuupr$$
$$gqqksngmjiplttgg$$
$$pjtltsqiftfnfsph$$

6. Now for all the 10000 ciphertexts we employed frequency analysis technique to get the frequency of each letter present in the ciphertexts.We used the code present in the file **vikrant_frequency_analysis.py** to employ the frequency analysis on the ciphertext.
7. From here we got the following results for the frequency analysis.
Letter Frequencies in 'alphabetical' order in ciphertext1:

f : 9920	g : 10025	h : 9913	i : 9897
j : 9937	k : 9898	l : 9909	m : 10080
n : 10114	o : 9995	p : 10068	q : 10123
r : 10087	s : 10054	t : 9979	u : 9985

8. Now from here we can clearly see that the letters are present in the ciphertext are only from **f to u**.
9. But to be double sure of our analysis that the letters present in the ciphertext text are indeed only from f to u we took another 10000 randomly generated plaintexts and stored them in **plaintext1_r.txt** file.
10. Now then we again using the same python script as above we firstly obtained the corressponding ciphertexts for the given plaintexts and then again employed the frequency analysis technique and got the frequency result as follows:
Letter Frequencies in 'alphabetical' order in ciphertext2:

$f : 9993$	$g : 9922$	$h : 9995$	$i : 10077$
$j : 9954$	$k : 10214$	$l : 9941$	$m : 10028$
$n : 9889$	$o : 10045$	$p : 9976$	$q : 9958$
$r : 9886$	$s : 10127$	$t : 10060$	$u : 9919$

Some Plaintexts from another set of randomly generated plaintexts:

$$\begin{gathered} rsgeldjidpkrqrff \\ rsgelljhdpjrqrff \\ hsmnofiggngfmqqm \\ hsmnonifgnffmqqm \\ jfrkooihqopogghh \\ jfrkogiiqoqogghh \end{gathered}$$

Corressponding Ciphertexts:

$$\begin{gathered} fnkigjhpolkiomfk \\ sfqkrlljqquqtnlk \\ uphsqihnkqhqhjls \\ iosgqlmihpjhlgpm \\ tspjpnmorungkunh \\ liksfjtmtftksnti \end{gathered}$$

11. Now from here we can clearly see that the generated ciphertexts are clearly see from the frequency analysis that the cipher text contains letters only from $\mathbf{f\ to\ u}$.
12. So from here we concluded that the letter range in the ciphertexts are from f to u.Also u is the 21th letter in abc.. And f is the 6th letter in abcd...Therefore 21-6+1=16 letters.Therefore we have got our 16 letters which are from f to u both inclusive which are the letters present in the ciphertext.
13. We already knew that the ciphertext contained only 16 letters and from above analysis we got the range of the letters which is f to u both inclusive.

Mapping

1. Now since we are knowing that the letters in ciphertext range from f to u. Also as per the hint provided by the spirit we are knowing that each letter is represented by 4 bits only and indeed we are having only 16 letters which is 2^4=>16 therefore we can represent each letter by 4 bits representing (0 to 15). The corresponding mapping for the letters is:

$$
\begin{gathered}
'f' :' 0000' \\
'g' :' 0001' \\
'h' :' 0010' \\
'i' :' 0011' \\
'j' :' 0100' \\
'k' :' 0101' \\
'l' :' 0110' \\
'm' :' 0111' \\
'n' :' 1000' \\
'o' :' 1001' \\
'p' :' 1010' \\
'q' :' 1011' \\
'r' :' 1100' \\
's' :' 1101' \\
't' :' 1110' \\
'u' :' 1111'
\end{gathered}
$$

2. Therefore we mapped each letter to its corresponding binary representation.

Breaking 6 Round DES

Generating Plaintext-Ciphertext Pairs

1. Now for breaking 6 round DES we make use of differential cryptanalysis.Now what we actually do in differential cryptanalysis is that we generate 5000 pairs

of plaintext and then we obtain their corresponding ciphertext.Therefore we would be having 5000 pairs of plaintext and their corresponding ciphertext.
2. We actually generate the $\mathbf{5000\ pairs}$ of plaintext using $\mathbf{vikrant\_gen\_plaintext.py}$ file.
3. We are actually using two 3 round characteristics both having a probability of $\mathbf{1/16 => 0.0625}$.But how are we getting this probability actually we are getting this probability by seeing the diagram 2 image(uploaded in answer 6) the probability is given by 1/4*1*1/4=1/16.Similarly by seeing the diagram 3 we would see that the probability is 1/4*1*1/4=1/16. For this probability we would be requiring approximately around $\mathbf{5000\ pairs\ which\ means\ total\ 10000}$ plaintexts are required for each of the characteristic and then we choose the key which occurred most often in them.
4. Each of the characteristic helps us to find the 30 bits of round 6 key. However, 3 of the S-boxes are common, so we only have 42 bits. The rest of the 14 bits can be found by exhaustive search.
5. The characteristics which we are going to use are(in hexademical):

$\mathbf{(4008000004000000)}$
$\mathbf{(0020000800000400)}$

6. We actually have to generate 5000 pairs of plaintext satisfying (4008000004000000) characteristic.We choose the plaintexts such that their input XOR after the Initial Permutation(IP) comes out to be (40 08 00 00 04 00 00 $00)_x$. Note that $P_1 \oplus P_2$ is not the same as characteristic, but $IP(P_1) \oplus IP(P_2)$ equals characteristic.

Therefore,
$\mathbf{(P1) \oplus (P2)\ should\ be\ (0000801000004000)}$
$\mathbf{IP(P_1) \oplus IP(P_2) = (4008000004000000)}$
$\mathbf{(P_1) \oplus (P_2) = (0000801000004000)}$

Here P_1 & P_2 are plaintext blocks.

7. Lets take an example:-
P_1=(D522D372E1BC322A)
P_2=(D5225362E1BC722A)
$(P_1)\oplus(P_2)$= (0000801000004000)
P_1 after Initial Permutation:
(1D6D211535FAA0CE)
P_2 after Initial Permutation:
(5D65211531FAA0CE)
$IP(P_1)\oplus IP(P_2)$=(4008000004000000)

8. Similarly now we generate have to generate 5000 pairs of plaintext satisfying (0020000800000400) characteristic.We choose the plaintexts such that their input XOR after the Initial Permutation(IP) comes out to be $(0020000800000400)_x$. Note that $P_1\oplus P_2$ is not the same as characteristic, but $IP(P_1)\oplus IP(P_2)$ equals characteristic.
Therefore,
$$\mathbf{(P_1)\oplus(P_2)\ should\ be\ (0000080100100000)}$$
$$\mathbf{IP(P_1)\oplus IP(P_2)=(0020000800000400)}$$
$$\mathbf{(P_1)\oplus(P_2)=(0000080100100000)}$$

Here P1 & P2 are plaintext blocks.

9. We made sure that the XOR of the input plaintext is (0000801000004000) for the first characteristic we ensured this thing using the python script present in **vikrant_gen_plaintext.py**. This Xor is obtained by applying inverse initial permutation to our characteristic xor which we know is (4008000004000000).
10. We have ensured that this property of XOR is maintained with the help of our python script present in the file **vikrant_gen_plaintext.py**.
Similarly for the second characteristic we made sure that

the XOR of the input plaintext is (0000080100100000) for the characteristic (0020000800000400) using the same python script.
11. But how are we actually doing this:
Actually what we are doing is that we are randomly generating and then selecting plaintext and once we have got a particular plaintext we perform its XOR with (0000801000004000) .As a result we obtain another plaintext actually now we know that the XOR of these two plaintexts would be (0000801000004000). We append both of them to our plaintext list.
12. After getting 5000 pairs of plaintexts we store these plain texts in file named plaintext1.txt. We can see the code in the file vikrant_gen_plaintext.py to get better idea. The heart of this code is the below lines:

PSEUDO CODE : –

```
XOR_value = list((bin(0x0000801000004000))[2:].zfill(64))
XOR_value = [int(x) for x in XOR_value]
binplaintexts = []
for i in range(5000):
   tmp = np.random.choice(['f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u'], size=(1, 16), replace=True)[0]
   bininput = []
   for i in tmp:
      bininput.extend([int(a) for a in char2hex[i]])
   binplaintexts.append(bininput)
   binplaintexts.append(list(np.bitwise_xor(bininput,
XOR_value)))
```

13. But how are we actually doing in this pseudo code:
Actually what we are doing is that we are randomly generating and then selecting plaintext and once we have got a particular plaintext we perform its XOR with (0000801000004000) .As a result we obtain another plaintext actually now we know that the XOR of these two plaintexts would be (0000801000004000) . We

append both of them to our plaintext list.In this way we would get the 5000 pairs of plaintext for the characteristic (0000801000004000) . These plaintexts are stored in plaintext1.txt.

14. Some Generated plaintexts stored in **plaintext1.txt** file are:-

$$\begin{gathered} uqqnjigoklmhhtgi \\ uqqnrifoklmhltgi \\ iqthknufnjsihhgi \\ iqthsntfnjsilhgi \\ jliulphiggkgktok \\ jliutpiiggkggtok \end{gathered}$$

15. Now similarly we would follow the same strategy to generate the 5000 plaintext pair for the characteristic (0020000800000400).We would store these plaintexts in **plaintext2.txt**file.

16. Some Generated plaintexts stored in **plaintext2.txt** file are:-

$$\begin{gathered} kmmqhtorslourimf \\ kmmqhlosslnurimf \\ ojglmjjtffprqolo \\ ojglmrjuffqrqolo \\ nqljqnknfuiggkhn \\ nqljqfkofuhggkhn \end{gathered}$$

17. Now we would be generating the ciphertexts for the plaintexts stored in plaintext1.txt file and plaintext2.txt file.For this we made use of pexcept library.

18. What actually we did is that with the help of python code present in the file **vikrant_final.py** we accessed the shell and logged in into the server game with the help of our credentials. We then executed the commands as given answer 2 in the exactly given sequence so as to reach the screen from where we can

do the chosen plaintext attack.
19. After this we one by one picked up all the 10000s plaintexts **plaintext1.txt** file and obtained their corresponding ciphertext and store the corresponding ciphertexts in the file **ciphertext1.txt** file.
20. Some of the obtained ciphertexts are:-

$$iiupffsqgngmkhli$$
$$julkgssgnnfmfrsm$$
$$ujptpguipkgimult$$
$$qossuqomjqoqllnk$$
$$grqprsqojnktlftu$$
$$hgsqklhfmrgojinh$$

21. Pseudo Code:

```
child = pexpect.spawn('ssh student@172.27.26.188')
child.expect('student@172.27.26.188\'s password:')
child.sendline('cs641')
child.expect('Enter your group name: ', timeout=1)
child.sendline("SecureSavants")
............(Some more Code)
f = open("plaintext1.txt", 'r')
f1= open("ciphertext1.txt",'w')

for line in f.readlines():
   child.sendline(line)
   print(child.before)
   f1.writelines(str(child.before)[48:64]+"\n")
```

22. Using the same technique we would be getting the **ciphertext2.txt** for the characteristic (0020000800000400) from the plaintexts obtained from **plaintext2.txt** file.
23. Some of the obtained ciphertexts from ciphertext2.txt file are:-

$$lnskhkrqmufqnnop$$
$$uonrhjgpgnonigfu$$
$$inintlgpltoqmujo$$
$$thnpsorporhugluk$$
$$sltkpjklonhuqmqi$$
$$sihrrijfmjgofkpn$$

24. In this way we would be getting the plaintext-ciphertext pairs.

Using The Ciphertexts obtained to break DES Finding input and Output XOR of the 6th rou

1. Now we starting taking the ciphertext from **ciphertext1.txt** file one by one and then we turn each letter into its corresponding binary corresponding binary representation using the mapping where f is 0000 and u is 1111.
2. Now we employed the inverse final permutation on the ciphertext to get (L6,R6) and (L'6,R'6) . L is the left half of the ciphertext and R is the right half of the ciphertext.C and C' represent the corresponding pairs of the ciphertextsBut we want to want to reach the output of the sbox of the 6th round to do this we are applying inverse final permutation on the ciphertext.
I have uploaded uploaded an image named Diagram 1 in answer 6 for better understanding.

3. Now we know to L6 is equal R5 (from the **diagram 1**).So now we use R5 and R'5 to find the output of the expansion box and thereby we would get the Xor of the input of the Sbox for the 6th round.
Let R5 and R'5 be the right half of the 5th round output. We know both R5 and R'5 since we know L6 and L'6 and R5=L6.
Therefore we get,

$\mathbf{R5 \oplus R'5}$

4. Now we find what would be the XOR after applying the expansion
Applying Expansion:

$\mathbf{E(R5) \oplus E(R'5)}$
Here E is the Expansion

5. Now we apply the 6th round key
Applying key:

$\mathbf{(E(R5) \oplus K6) \oplus (E(R'5) \oplus K6)}$
$\mathbf{E(R5) \oplus E(R'5)}$

=>This is the input XOR of the sbox
We have written all these steps in form of code to compute to input of the sbox.In this way we would calculate the input of the sbox.
6. Now once we have found out the input XOR of the sbox now we have to find out output XOR of the sbox.Lets see how we can get the output XOR of the sbox.
7. To get the output XOR of the sbox what we firstly do is we can see from the diagram 1 that we already know R6.Now in the diagram 1 we can see that output of the sbox is actually nothing but the XOR of R6 and L5.

8. From $\mathbf{diagram\ 2}$ image (uploaded in answer 6) we can see that the value of L5 is (04000000) for the first characteristic and from diagram 3 (uploaded in answer 6) we can see that the value of L5 is (00000400) for the second characteristic.
9. Using these values now we would find the value of output XOR using the equation $\mathbf{L5 \oplus (R6 \oplus R'6)}$.But we can see in the $\mathbf{diagram\ 1}$ that there is a permutation also present just after the output of the sbox therefore we also have to apply inverse

permutation to get the the out of the sbox.
10. In this way we obtained the XOR of the output of the sbox of the 6th round.
11. Now we have XOR for both input and output of the sbox of the 6th round DES.Now we can further apply our analysis.
Now Let:

$$E(R5) = \alpha_1\alpha_2\cdots\alpha_8 \quad and \quad E(R\prime 5) = \alpha'_1\alpha\prime_2\cdots\alpha'_8 \quad where$$

$$\mid \alpha_i \mid = 6 = \mid \alpha'_i \mid \quad and$$

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

$$\beta_i = \alpha_i \oplus k_{6,i} \quad and \quad \beta'_i = \alpha'_i \oplus k_{6,i}$$

$$S(\beta_i) \oplus S\text{'}(\beta_i) = \gamma_i \oplus \gamma'_i$$

$$At \quad this \quad point, \quad we \quad know$$

$$\alpha_i, \alpha'_i, \beta_i \oplus \beta'_i \quad and \quad \gamma_i \oplus \gamma'_i$$

$$k_6 = k_{6,1}\, k_{6,2}\cdots k_6, 8 \quad and$$

$$\beta_i = \alpha_i \oplus k6, i \quad and \quad \beta'_i = \alpha'_i \oplus k_6, i$$

$$S(\beta_i) \oplus S\prime(\beta'_i) = \gamma_i \oplus \gamma'_i$$

$$\alpha_i, \alpha'_i, \beta_i \oplus \beta'_i\prime \quad and \quad \gamma_i \oplus \gamma'_i$$

After this then we created 8 $*$ 64 key matrix to store the number of times a key k $\in$ [1, 64] satisfies the possibility of being a key to Si box, where i $\in$ [1,8]

We find the set

$X_i = \beta, \beta\prime \mid \beta \oplus \beta\prime = \beta_i \oplus \beta_i' \quad and \quad S_i(\beta) \oplus S_i(\beta) = \gamma$

Then for each k ∈ [1, 64], we check whether α_i ⊕k=β and (β,β′)∈X_i for some β′

On satisfying the above condition for Si box,the key[i][k] will be incremented by 1.We are using this to calculate the frequencies of a particular 6-bit key for each sbox.

12. Now as taught in class we know that for the 1st characteristic our input XORs for five Sboxes (S2,S5,S6,S7,S8) are zero therefore their output XORs are zeros.Similarly for 2nd characteristic input XORs for five Sboxes(S1,S2,S4,S5 and S6) are zero so their output XORs are zero.Now therefore for the first sbox we find the most frequent 6 bit key corresponding to the sboxes {S2, S5, S6, S7, S8} and Sboxes {S1,S2,S4,S5,S6} for 2nd characteristics

13. Performing the above steps for the characteristic (4008000004000000) is that we get the partial key using S2,S5,S6,S7,S8 as (59,37,50,14,61) as the input to these Sboxes is 0 in round 4.

14. By using the steps and analysis described above we get the following result:

$S-box$	Max	$Mean$	Key
$S1$	626.0	330	45
$S2$	1543.0	390	59
$S3$	600.0	333	37
$S4$	538.0	335	7
$S5$	798.0	359	37
$S6$	1467.0	379	50
$S7$	914.0	353	14
$S8$	893.0	359	61

15. Now Performing the same steps for the characteristic (0020000800000400) is that we get the

partial key using S1,S2,S4,S5,S6 as (45,59,7,37,50) as the input to these Sboxes is 0 in round 4.We know that S2,S5,S6 are between first characteristic and second characteristic.

$S-box$	Max	$Mean$	Key
$S1$	782.0	338	45
$S2$	802.0	347	59
$S3$	641.0	328	37
$S4$	1491.0	402	7
$S5$	856.0	349	37
$S6$	1477.0	380	50
$S7$	642.0	340	14
$S8$	523.0	327	61

16. Now as taught in class we know that for the 1st characteristic our input XORs for five Sboxes (S2,S5,S6,S7,S8) are zero therefore their output XORs are zeros.Similarly for 2nd characteristic input XORs for five Sboxes(S1,S2,S4,S5 and S6) are zero so their output XORs are zero.Now therefore for the first sbox we find the most frequent 6 bit key corresponding to the sboxes {S2, S5, S6, S7, S8} and Sboxes {S1,S2,S4,S5,S6} for 2nd characteristics.
17. From here we can see that S3 can never be Zero.
18. Using the above analysis of characteristics we get 42 bits of key 7*6=>42bits out 56 bits of key has been found.The 48 bit key is:
101101111011XXXXXX000111100101110010

19. We have put XXXXXX in place of key from the sbox S3 because S3 can never be zero therefore we put XXXXXX in place of it.

Brute Force to search for remaining bits of the

1. Actually in key scheduler we actually have initially a 64 bit key.This key is then passed to Permuted choice

1(PC1) which converts this 64 bits key to 56 bits of key.After this we pass the 56 bit key to Permuted choice 2(PC2) to obtain the 48 bit key.This key is actually used in rounds.There is one specific shifting operation which is specific to different different rounds.
2. Now to the obtained 48 bit key to convert it into 56 bit key we apply key scheduler Permuted choice 2(PC2),we get the following output:-

X11XX1XX01011X100XX11X11100X000100

3. Now to find the remaining key bits we apply brute force method.In the above 56 bit we can see that there are 14 Xs and for each X there are 2 possibilities 0 or 1 therefore. We have 2^{14} possibilities.
4. Actually what we did is that we passed the plaintext **’fghijklmnopqrstu’** to the server we get the output ciphertext as **“gqmpugjhnunpujko”**.Now with each possible key we encrypted the plaintext 'fghijklmnopqrstu’ with that particular key and then then we check if the resulting ciphertext text is equal to “gqmpugjhnunpujko”.If for a key we obtained this ciphertext then that key is identified as our required key.In this way we were able to find the complete key.
5. The actual 56 bit key is:
0110111001011110011110111000000100001101
6. Now using this 56 bit key we derived the various round keys we found all the round keys using the function roundkey_des function present in the code **vikrant_final_break_DES.ipynb**.Actually here in this round key function we are actually implementing the key scheduler algorithm since now we are already having the 56 bit key we can now generate all the round key using this key.
7. The various keys which we get as out are:

The key is 0110111001011110011110111000000
Round 0 key is 11111100010011110000011111

Round 1 key is 01101111001111110110001010
Round 2 key is 11101010111101001110110101
Round 3 key is 11011001110001110101101010
Round 4 key is 01100100110110111011101111
Round 5 key is 10110111101110010100011110

Password Decryption

1. Actually to clear the level the spirit told us that if we entered the command password then we would get the password in coded format and with the help of the key which we find out we have to decode this ciphertext to get our password to clear this level. When we entered the command password we got **"fpljsgojpkhukunguknt ppkngjfsjgkm"** as our coded password.
2. Now we firstly converted "fpljsgojpkhukunguknt ppkngjfsjgkm" into binary and then to decimal values and then we divided it two parts as DES only works on 8 bytes of plaintext at a time we get the output as {10,100,209,148,165,47,95,129} and {245,142,170,88,20,13,65,87}.
3. OUTPUT:-

fp	10
lj	100
sg	209
oj	148
pk	165
hu	47
ku	95
ng	129
uk	245
nt	142
pp	170
kn	88
gj	20
fs	13
jg	65
km	87

Here each block is of 8 bytes.

4. Now we pass these values in vikrant_des.cpp file.We get the output of decryption as: **skwvjxdsgq000000**
5. We thought that at last the 000000 would be just for padding purposes and therefore we tried without using the 000000.
6. We tried "skwvjxdsgq" without quotes as the password and we successfully cleared the level.
7. Therefore our required password used to clear this level is **"skwvjxdsgq"** without the quotes.This is our final command to clear this level.

References:
1. For Understanding breaking of 6 round DES:
-> Lecture Notes Of CS641
->https://medium.com/@jnaman806/breaking-des-using-differential-cryptanalysis-958e8118ff41
2. For Diagrams 1,2,3:
->Research Gate
3.For Understanding Key Scheduling:

->https://ritul-patidar.medium.com/key-expansion-function-and-key-schedule-of-des-data-encryption-standard-algorithm-1bfc7476157

Q5 Password
5 Points

What was the password used to clear this level?

The Password used to clear this level is:-

skwvjxdsgq

Q6 Code
0 Points

Please add your code here. It is MANDATORY.

FINAL_SECURESAVANTS_ASSIGN4.zip Download

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

Assignment 4

● Graded

Select each question to review feedback and grading details.

Group

Vikrant Chauhan
VISHAL KUMAR
Souvik Mukherjee
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	**Resolved** **80** / 80 pts
Question 5	
Password	**5** / 5 pts
Question 6	
Code	**0** / 0 pts