Jeremy Barthélemy  
ECE 646  
Hw. 4

**Problem 7**Compute the output L1R1 of the first round of DES encryption, assuming that the input plaintext block consists of the sequence F0F0F0F0F0F0F0F0, and the key (including odd parity bits) consists of the sequence 0E0E0E0E0E0E0E0E.L1 = R0  
R1 = L0 XOR f(R0, K1)   
First convert to binary:  
F0F0F0F0F0F0F0F0 = 1111000011110000111100001111000011110000111100001111000011110000  
0E0E0E0E0E0E0E0E = 0000111000001110000011100000111000001110000011100000111000001110  
We need to perform the initial permutation upon the plaintext block first.  
  
From here, we convert this table:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

To the table below, by performing the initial permutation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Which is FFFF0000FFFF0000 in hex.  
We can take the right 4 bytes (which is coincidentally the same as the left 4) and set L0 to this value.  
L1 = FFFF0000  
R1 is harder to calculate.  
The next step is to perform the expansion permutation upon R­­­0:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |

Now, with the key(0E0E0E0E0E0E0E0E), we ignore every eighth bit, and perform the Permuted Choice One. With this, we build two tables – one representing C0 and the other D0 (the first and second ones, respectively).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

C0 = 0000000000000000000000000000  
D0 = 1111111111111111111111110000  
We must treat these two segments of 28 bits as separate values for now, by performing circular left shifts. For round one, we simply perform 1 left rotation, so C0 and D0 become:  
C0 = 0000000000000000000000000000  
D0 = 1111111111111111111111100001  
These are then recombined:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

and fed into Permuted Choice 2:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

Now we have the 48 bits necessary to XOR the expanded R0 with:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

Now, we will perform the S-box Substitution Operations:

S1: 011111  
S2: 111111  
S3: 111111  
S4: 111110  
S5: 001110  
S6: 111111  
S7: 111110  
S8: 111110  
Now, we look up in each S-box the 4 bit output:  
S1: 8 = 1000  
S2: 9 = 1001  
S3: 12 = 1100  
S4: 4 = 0100  
S5: 6 = 0110  
S6: 13 = 1101  
S7: 2 = 0010  
S8: 8 = 1000  
Put it together and we have  
10001001110001000110110100101000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

Next, we perform another permutation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Finally, we XOR these 32 bits with L0 to get R1:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

L1 = FFFF0000  
R1 = 11100111011100100111010110001000 = E7727588  
  
  
**Problem 8**

Compute the bits number 1, 32, 33, and 64 at the output of the first round of the DES encryption, assuming that the plaintext block is equal to M=”C000 0000 0000 0003”, and the round key K1 is equal to “FEDC BA98 7654” (all values in the hexadecimal notation).L1R1(1) = L1(1) = R0(1)  
L1R1(32) = L1(32) = R0(32)  
L1R1(33) = R1(1)  
L1R1(64) = R1(32)   
IP(M) = 01 00 00 80 01 00 00 80  
L0 = R0 = 01 00 00 80  
L1 = R0 = 01 00 00 80 🡪   
L1(1) = 0  
L1(32) = 0

R1(1) = L0(1) XOR P1(1) = P1(1) = Sboxes(16) = S4(4)  
R1(32) = L0(32) XOR P1(32) = Sboxes(25) = S7(1)  
Means that X1 bits (output from first XOR) that we find interesting for this case are the inputs to boxes S4 and S7, which are X1(19...24) and X1(37…42).

We expand R0 first to:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

We only need to XOR rows 4 and 7 from the round key actually to find the inputs to our S-boxes:  
Input to S4 = 111010  
Input to S7 = 001001  
Output from S4 = 2 = 0010  
Output from S7 = 0 = 0100  
We are interested in bit 4 of S4, which is 0, and bit 4 of S4, which is also 0.   
L1R1(33) = 0  
L1R1(64) = 0  
So all of the bits requested are 0.  
  
**Problem 9 (5 points)**  
**Suppose the DES f function mapped every 32-bit input R, regardless of the value of the 48-bit input K, to   
a)  value of R,**For this case, we simply have Li = Ri-1 and Ri = Li-1 XOR Ri-1.  
What would be the expression for R16L16 as a function of L0R0 during encryption?1.   
L1 = R0  
R1 = L0 XOR R0  
L2 = L0 XOR R0  
R2 = R0 XOR R0 XOR L0 = 0 XOR L0 = L0  
L3 = L0  
R3 = L0 XOR L0 XOR R0 = R0  
L4 = R0  
R4 = L0 XOR R0  
L5 = L0 XOR R0  
R5 = L0 XOR R0 XOR R0 = L0  
L6 = L0  
R6 = L0 XOR L0 XOR R0 = R0  
L7 = R0  
R7 = L0 XOR R0  
L8 = L0 XOR R0  
R8 = R0 XOR L0 XOR R0 = L0  
L9 = L0  
R9 = L0 XOR R0 XOR L0 = R0  
L10 = R0  
R10 = R0 XOR L0  
…**L16 = R0  
R16 = R0 XOR L0**  
  
2. **What would be the expression for L0R0 as a function of R16L16 during decryption?**  
R15 = L16  
L15 = L16 XOR R16 = L12 = L9 = L6­ = L3 = L0  
R14= L16 XOR R16  
L14 = L16 XOR L16 XOR R16 = R16  
R13= R16  
L13 = L16  
R12 = L16  
L12 = L16 XOR R16  
R11 = L16 XOR R16  
L11 = R16

…  
**R0 = L16  
L0 = L16 XOR R16**  
  
**b) 16 leftmost bits of R concatenated with the bitwise-complement of 16 rightmost bits of R, i.e., if R = RL || RR, then F(R, K) = RL || bitwise\_complement\_of (RR).**Li = Ri-1 and Ri = L(Ri-1) || BC\_R(Ri-1).  
1. **What would be the expression for R16L16 as a function of L0R0 during encryption?**  
L1 = R0  
R1 = L0 XOR f(R0) = (L(L0 XOR R0) || BC\_R(L0 XOR R0))  
L2 = (L(L0 XOR R0) || BC\_R(L0 XOR R0))  
R2 = R0 XOR f((L(L0 XOR R0) || BC\_R(L0 XOR R0))) = R0 XOR (L(L0 XOR R0) || R(L0 XOR R0)) = (L(L0) || R(L0)) = L0  
L3 = (L(L0) || R(L0)) = L0  
R3 = (L(L0 XOR R0) || BC\_R(L0 XOR R0)) XOR f((L(L0) || R(L0)) =  
(L(L0 XOR R0) || BC\_R(L0 XOR R0)) XOR (L(L0) || BC\_R(L0))   
= L0 XOR (L(R0) || BC\_R(R0)) XOR (L(L0) || BC\_R(L0)) = (L(R0) || BC\_R(R0)) XOR (0 || 1) =  
L(R0) || R(R0) = R0  
L4 = R0  
R4 = L0 XOR f(R0) = L0 XOR (L(R0) || BC\_R(R0))  
**…  
L16 = R0  
R16 = L0 XOR (L(R0) || BC\_R(R0))**  
2. **What would be the expression for L0R0 as a function of R16L16 during decryption?**   
R15 = L16  
L15 = R16 XOR f(L16) = R16 XOR (L(L16) || BC\_R(L16))  
R14= R16 XOR (L(L16) || BC\_R(L16))  
L14 = L16 XOR f(R16 XOR (L(L16) || BC\_R(L16))) = L16 XOR R16 XOR L16= R16   
R13 = R16  
L13 = R16 XOR (L(L16) || BC\_R(L16)) XOR f(R16) =   
R16 XOR (L(L16) || BC\_R(L16)) XOR (L(R16) || BC\_R(R16)) = 0|| 1 XOR (L(L16) || BC\_R(L16)) = L16R12 = L16  
L12 = R16 XOR f(L16) = R16 XOR (L(L16) || BC\_R(R16))  
…  
**R0 = L16  
L0 = R16 XOR f(L16) = R16 XOR (L(L16) || BC\_R(R16))   
  
  
  
  
Problem 10 (3 bonus points)**

How many blocks of the ciphertext are needed in the exhaustive search ciphertext-only attack against DES in order to make the probability that the attack ends with finding an incorrect key smaller than 1% under the assumption that the plaintext consists of only 64 out of 256 ASCII characters?

256 possible keys, only one is correct (i.e. 256 – 1 incorrect keys)  
8 characters applied per block (each character is 8 bits)   
  
With 64 ASCII characters:   
From 240 invalid keys, probability to get under 1%,   
1-n240-64 <= 0.01,   
n2-24 = 0.99,   
0.99(2^24)   
=   
16609448 bits -->  
2076181 bytes 🡪 approximately 259523 ciphertext blocks required