Problem 6- Two Round DES find K1

February 27, 2020

[1]:
```
import sys
sys.path.append('../')
import crypto_utils as utils
```

[2]:
```
M = '1110100101110101001110101101010101110111010111010111110010010100'
C = '1011111111100000101001000111111010000000010011101011110110110001'
L0 = M[:32]
R0 = M[32:]
L2 = C[:32]
R2 = C[32:]
```

[3]:
```
L2_xor_L0 = utils.xor(L2,L0)
```

[4]:
```
# So what I want to calculate is K1 from equation L2 xor L0 = f(R0, K1)
# Need to do E(R0). Also need to do invS(invP(L2 xor L0))
# Then will have E(R0) xor K1 = invS(invP(L2 xor L0))
```

[5]:
```
# E(R0), will only need first six bits (thanks Dave for not making us do the␣
 ↪whole thing)
E_R0 = [R0[-1], R0[0], R0[1], R0[2], R0[3], R0[4]]
print(E_R0)
```

```
['0', '0', '1', '1', '1', '0']
```

[6]:
```
# Doing the inverse P, I will just look in the table for the 1,2,3,4 of P
# Those are located at 8, 16, 22, 30 (0 indexed)
invP = [L2_xor_L0[8], L2_xor_L0[16], L2_xor_L0[22], L2_xor_L0[30]]
print(invP)
```

```
['1', '1', '1', '1']
```

[7]:
```
# invP(L2 xor L0) = (1111)_2 = 15 = output of S1
# S1 possibilities are (0,5), (1,1), (2,8), (3,0)
# (0,5) = (00)_2, (0101)_2 = 001010
# (1,1) = (01)_2, (0001)_2 = 000011
# (2,8) = (10)_2, (1000)_2 = 110000
# (3,0) = (11)_2, (0000)_2 = 100001
```

```python
# Those are the possible strings before S1. One of those is equal to E(R0) xor K1
# Therefore to get the possibilities for K1, I just need to xor each possibility with E(R0)
print("Possibility 1: " + ''.join(utils.xor('001010', E_R0)))
print("Possibility 2: " + ''.join(utils.xor('000011', E_R0)))
print("Possibility 3: " + ''.join(utils.xor('110000', E_R0)))
print("Possibility 4: " + ''.join(utils.xor('100001', E_R0)))
```

```
Possibility 1: 000100
Possibility 2: 001101
Possibility 3: 111110
Possibility 4: 101111
```

[8]:

```python
# The above output is the possibilities for K1
```