In [1]:

```
# S box
sbox = [9, 4, 10, 11, 13, 1, 8, 5, 6, 2, 0, 3, 12, 14, 15, 7]


def convertNumToAsciiBit(x): # coverts decimal to binary
    y = ""
    for i in range(len(x)):
        val = ord(x[i])
        j = 7
        ans = ""
        while j >= 0:
            w = val // (pow(2, j))
            ans += str(w)
            val = val % (pow(2, j))
            j -= 1
        y += ans
    return y


def convertAsciiToChar(x): # converts ASCII value to char
    y = ""
    for i in range(0, len(x), 8):
        ans = 0
        for j in range(8):
            ans += int(x[i + j]) * pow(2, 7 - j)
        if i == len(x) - 8:
            if chr(ans) != '#':
                y += chr(ans)
        else:
            y += chr(ans)
    return y


def keyExpansion(key): # generates 2 round keys
    x = [key[:4], key[4:8], key[8:12], key[12:16]]
    for i in range(4): # binary to decimal for each nible
        x[i] = list(map(int, x[i]))
        x[i] = x[i][0] * 8 + x[i][1] * 4 + x[i][2] * 2 + x[i][3]
    keylist = [x[0], x[1], x[2], x[3]]
    for i in range(2):
        w2 = [0, 0, 0, 0]
        if i == 0:
            val = 8 # rcon for first round
        else:
            val = 3 # rcon for 2nd round
        w2[0] = keylist[4 * i] ^ val ^ (sbox[keylist[4 * i + 3]])
        w2[1] = keylist[4 * i + 1] ^ 0 ^ (sbox[keylist[4 * i + 2]])
        w2[2] = w2[0] ^ keylist[4 * i + 2]
        w2[3] = w2[1] ^ keylist[4 * i + 3]
        keylist.append(w2[0])
        keylist.append(w2[1])
        keylist.append(w2[2])
        keylist.append(w2[3])
    return keylist # has all 3 sub-keys


def getByteFromBit(x):# converts binary to bytes
    y = []
    i = 0
    while i < (len(x)):
        y.append(8 * x[i] + 4 * x[i + 1] + 2 * x[i + 2] + x[i + 3])
        i += 4
    return y


def mixCols(y): # applies Mix-Columns
    w = []
    for i in range(len(y)):
        val = y[i] * 4
        if val >= 32:
            val ^= 38
        if val >= 16:
            val ^= 19
        w.append(val)
    ans = [0, 0, 0, 0]
    ans[0] = y[0] ^ w[1]
    ans[1] = y[1] ^ w[0]
    ans[2] = y[2] ^ w[3]
    ans[3] = y[3] ^ w[2]
    return ans


def convertByteToBit(y): # converts byte value to binary
    cipher = []
    for i in range(len(y)):
        val = y[i]
        val1 = val // 8
        cipher.append(val1)
        val = val % 8
        val1 = val // 4
        cipher.append(val1)
        val = val % 4
        val1 = val // 2
        cipher.append(val1)
        val1 = val % 2
        cipher.append(val1)
    cipher = list(map(str, cipher))
    return "".join(cipher)


def mult(x, y):
    val = x * y
    if y == 2:
        if val >= 32:
            val ^= 38
        if val >= 16:
            val ^= 19
    else:
        val = x * 8
        if val >= 64:
            val ^= 76
        if val >= 32:
            val ^= 38
        if val >= 16:
            val ^= 19
        val ^= x
    return val


def inverseMixCols(y): # applies inverse Mix-Columns
    w = [0, 0, 0, 0]
    w[0] = mult(y[0], 9) ^ mult(y[1], 2)
    w[1] = mult(y[1], 9) ^ mult(y[0], 2)
    w[2] = mult(y[2], 9) ^ mult(y[3], 2)
    w[3] = mult(y[3], 9) ^ mult(y[2], 2)
    return w


def aesDecrypt(y, keylist): # applies Decryption Algorithm
    j = 2
    for i in range(len(y)):
        y[i] ^= keylist[4 * j + i]
    j = 1
    while j >= 0:
        y[1], y[3] = y[3], y[1]
        for i in range(len(y)):
            y[i] = sbox.index(y[i])
        for i in range(len(y)):
            y[i] ^= keylist[4 * j + i]
```

```python
        if j != 0:
            y = inverseMixCols(y)
        j -= 1
    return convertByteToBit(y)


def aesEncrypt(y, keylist): # applies Encryption Algorithm
    for i in range(len(y)):
        y[i] ^= keylist[i % 4]
    for i in range(1, 3):
        for j in range(len(y)):
            y[j] = sbox[y[j]]
        y[1], y[3] = y[3], y[1]
        if i != 2:
            y = mixCols(y)
        for j in range(len(y)):
            y[j] = y[j] ^ keylist[4 * i + j]
    return convertByteToBit(y)


if __name__ == "__main__":
    print("Enter the plaintext (Please enter text in inverted commas, eg : \"abcde\"): ")
# any length char input
    x = input()
    print("Enter the key : ") # char input of length 2
    key = input()
    if len(key) != 2:
        print("BAD KEY : Should be 16 bits")
        exit(0)
    key = convertNumToAsciiBit(key)
    keylist = keyExpansion(key)
    if len(x) % 2 != 0:
        x += '#'  # filler - #
    x = convertNumToAsciiBit(x)
    x = list(map(int, x))
    i = 0
    cipher = ""
    while i < len(x) - 1:
        y = getByteFromBit(x[i:i + 16])
        cipher += aesEncrypt(y, keylist)
        i += 16
    print("Cipher text after encryption is : ")
    print(cipher)
    print(convertAsciiToChar(cipher))
    x = list(map(int, cipher))
    i = 0
    plaintext = ""
    while i < len(x) - 1:
        y = getByteFromBit(x[i:i + 16])
        plaintext += aesDecrypt(y, keylist)
        i += 16
    print("Plain text after decryption is : ")
    print(plaintext)
    print(convertAsciiToChar(plaintext))
```

```
Enter the plaintext (Please enter text in inverted commas, eg : "abcde"):
Hello I am Jessica
Enter the key :
10
Cipher text after encryption is :
10100010011010001001011111101110010101101100000100001001111110011110101110101101100111000
1010110001101101110010101000011111101101000000111110100
¢hîVÁ ùëV6åCöô
Plain text after decryption is :
01001000011001010110110001101100011011110010000001001001001000000110000101101101001000000
1001010011001010111001101110011011010010110001101100001
Hello I am Jessica
```

In [ ]: