

HW1 writeup

109550127 宋哲頤

(HW)LSB:

從 code 可以知道解密系統會 print 出明文模 3 之後的內容，也就是 LSB。在講義裡面說已知 a_0 可以推出 a_1 再推出 a_2到全部。只是這裡從 %2 變 %3，因此將 $(\text{pow}(3,-e)*c) \% n$ 的密文傳給系統會回傳 $\text{pow}(3,-1)*m$ 的明文。

- Oracle
 - $(2^{-1})^e c \rightarrow 2^{-1}m$
- Inference
 - | | | |
|-------|-------|-------|
| y_2 | x_1 | x_0 |
|-------|-------|-------|
 - $2^{-1}m = 2y_2 + x_1 + 2^{-1}x_0$
 - $r = [2y_2 + x_1 + 2^{-1}x_0]_{\text{mod } n} \pmod{2}$
 $= [2^{-1}x_0]_{\text{mod } n} + x_1 \pmod{2}$
 - $\Rightarrow x_1 = r - [2^{-1}x_0]_{\text{mod } n} \pmod{2}$

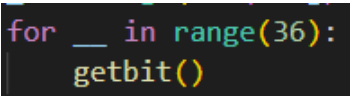
```
while True:
    inv = pow(3,-1,n)
    cc = (c * pow(inv,e*i,n)) % n
    p.sendline(str(cc))
    m = p.recvline().strip()[:]
    print(m)
    m = int(m)
    nexta = (m - (a*inv) % n) % 3
    print ("Number:" + str(i))
    if nexta == 0:
        cnt += 1
        if cnt == 10:
            break
    else:
        cnt = 0
    a = a*inv + nexta
    plaintext = 3**i*nexta + plaintext
    print (long_to_bytes(plaintext))
    i += 1
print (long_to_bytes(plaintext))
```

此為實作以上解法的 code。

另外 cnt 是作為計算多項式的參數是否都得出了，我設定 10 來推測經歷 10 次參數為 0 可以代表已經完成了。

(HW) XOR-revenge:

一開始以為是上課教的Fibonacci LFSR，仔細看發現XOR的順序不對，上網查後發現是galois LFSR。這題我是用上課提到的companion function進行運算。已知original state為64bits，且從0xda785fc480000001可知他的polynomial function。以未被改變的70個bits中的64bits去和companion matrix結合就能產生64個聯立方程式解initial state。

因為  的關係，每shift 67次才會append到

output上，因原來output長度為406再減70後可知在37*336個state之後會開始出現沒被XOR的state。

$$\begin{bmatrix} P(0) & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ P(63) & 0 & \cdots & 0 & 0 \end{bmatrix}^{36+37*I}$$
 的第一行去乘上 original state)就會等於

$state_{(36+37*I)-0}$ 之題目給的未受干擾的 output，之後就是透過高斯消去法解出 64bits 的 original state，來透過 getbit()去 xor 原本被污染的 336 個 output，還原得到 flag。

Code:

製作 companion matrix

```

poly=0xda785fc480000001
comatrix=np.ndarray((64,64),dtype=int)
for i in range(64):
    for j in range(64):
        if j==0:
            if((poly)&(1 << (63-i))):
                comatrix[i][j]= 1
            else:
                comatrix[i][j]= 0
        elif j==i+1:
            comatrix[i][j]=1
        else:
            comatrix[i][j]=0

```

把64個聯立方程式的係數放到equationMatrix裡面

```

trans37= comatrix.copy()

for i in range(36):
    trans37=trans37.dot(comatrix)%2
tmp=np.eye(64,64,dtype=int)
for i in range(36):
    tmp=tmp.dot(comatrix)%2
print(tmp)
equationMatrix=np.eye(64,64,dtype=int)
for i in range(336):
    tmp=tmp.dot(trans37)%2
for i in range(64):
    equationMatrix[i]=tmp[0]
    tmp=tmp.dot(trans37)%2

```

高斯消去法解出initial state

```

ansvec=outpt[336:336+64]
for i in range(64):
    col=0
    for j in range(64):
        if equationMatrix[i][j] ==1:
            col=j
            break

```

```

    for k in range(64):
        if k==i or equationMatrix[k][col]==0:
            continue
        for j in range(64):
            equationMatrix[k][j]^=equationMatrix[i][j]
        ansvec[k]^=ansvec[i]
ostate =equationMatrix.T.dot(ansvec)%2

```

最後去 xor initial state 解出 flag

```

for i in range(336):
    for _ in range(36):
        getbit()
    output[i] ^= getbit()

flag=b''

ans=[]
for i in range(42):
    c=int(''.join(map(str,output[i*8:(i+1)*8])),2)
    ans.append(col)
flag=bytes(ans)

print(flag)

```

(LAB)COR

```

vector<unsigned int> get_state(unsigned int tap,uint size){
    int output[200];
    vector<unsigned int> vec;
    for(unsigned int state = 0;state < 1ll << size;state++){
        LFSR lfsr = LFSR(state,tap,size);
        for (int i = 0; i < 232; i++){
            lfsr.getbit();
        }
        for(int i=0;i<200;i++){
            output[i]=lfsr.getbit();
        }
        double cor = cal_cor(output, &result[LEN(result) - 200]);
        if(cor >= 0.7){
            vec.push_back(state);
        }
    }
    return vec;
}

```

我依照上課的步驟先枚舉 state 產生 200bits，在看有幾成的機率一樣，先預設 0.7 一樣就當作可以，這樣就能獲得 lfsr 2 跟 3 的 state。

```
for(unsigned int state1 : states[0]){
    for(unsigned int state2 : states[1]){
        for(unsigned int state0 = 0; state0 < 111 <<size[0];state0++){
            LFSR lfsr0 = LFSR(state0, taps[0], size[0]);
            LFSR lfsr1 = LFSR(state1, taps[1], size[1]);
            LFSR lfsr2 = LFSR(state2, taps[2], size[2]);
            int x0,x1,x2;
            for (int i = 0; i < 232; i++){
                x0=lfsr0.getbit();
                x1=lfsr1.getbit();
                x2=lfsr2.getbit();
            }
            for(int i=0;i<200;i++){
                x0=lfsr0.getbit();
                x1=lfsr1.getbit();
                x2=lfsr2.getbit();
                output[i] = x0 ? x1 : x2;
            }
            double cor = cal_cor(output, &result[LEN(result) - 200]);
            if(cor >= 1){
                origin_states[0] = state0;
                origin_states[1] = state1;
                origin_states[2] = state2;
                goto final;
            }
        }
    }
}
```

最後再用這兩個得出正確的 lfsr 1。

```
final:
    LFSR lfsr0 = LFSR(origin_states[0], taps[0], size[0]);
    LFSR lfsr1 = LFSR(origin_states[1], taps[1], size[1]);
    LFSR lfsr2 = LFSR(origin_states[2], taps[2], size[2]);
    for(int i=0;i<29;i++){
        for(int j=0;j<8;j++){
            int x0,x1,x2,o;
            x0 = lfsr0.getbit();
            x1 = lfsr1.getbit();
            x2 = lfsr2.getbit();
            o = x0 ? x1 :x2;
            flag[i] |= (result[i*8+j] ^ o) << (7-j);
        }
    }
    cout<<flag;
}
```

將得出的三個 lfsr 推回去做。

(LAB)POA:

先對每一個 block 的第 15 個 byte 到第 0 個 byte 設定 postfix 為已知後綴明文和 prefix 的密文，但在做

```
now=prefix+bytes([i^last_ct[idx]])+posfix+block_ct
```

時，不太懂講者所說的 postfix 設為 0，然後找 0x80，因此這題目前還不太懂。