Homework 0x01

1. nLFSR

Problem

這題給了一個 64-bit 的 LFSR, 比較不一樣的是這個 LFSR 每 43 步才會產出一個 bit
而在題目裡要去預測 LFSR 的下一個 bit，答對加分答錯扣分，分數到一個值後可以拿到 Flag。

Concept

雖然 LFSR 每 43 步才會產一個 bit，但這也可以看做是一種 LFSR。而因為 43 跟狀態數 2^{64} 互質，因此這個 LFSR 最長可能的狀態數也是 2^{64}。

一開始我使用上課教的方式去湊 64 個等式出來，並製作出如下圖的方程式。然而湊完後要解方程式等式卻發現因為 rank 不夠而無法解。

$$\begin{bmatrix} C^{42}[1st\ row] \\ C^{85}[1st\ row] \\ C^{128}[1st\ row] \\ \cdots \\ C^{2751}[1st\ row] \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \cdots \\ a_{63} \end{bmatrix} = \begin{bmatrix} a_{42} \\ a_{85} \\ a_{128} \\ \cdots \\ a_{2751} \end{bmatrix}, \text{ where } \quad \text{C=} \begin{bmatrix} c_0 & 1 & 0 & 0 \ldots 0 \\ c_1 & 0 & 1 & 0 \ldots 0 \\ c_2 & 0 & 0 & 1 \ldots 0 \\ c_3 & 0 & 0 & 0 \ldots 0 \\ \cdots & \cdots & \cdots & \\ c_{63} & 0 & 0 & 0 \ldots 1 \end{bmatrix}$$

當時沒有想到好解法後，因此就暫時放棄尋找另解。(後來有想到其實可以多做幾個等式，並把多餘的 row 刪掉一樣可以解)

之後又查到了一個演算法 berlekamp massey。
這個演算法給定 2*n(bit) 的 LFSR output，在這題中就是 128 個 LFSR 的 output 之後，便可以去計算這個 LFSR 的 parameter $P(x)$。因此我們只要成功從題目中拿到 128 bit 的 output，就可以使用此演算法去計算 P(x)，進而去計算 Companion matrix 並成功預測接下來的每個 bit，成功拿到 Flag。

Note: 題目中若使用隨機策略亂猜，每次猜的分數的期望值為 $E(x) = 0.02 * 0.5 + (-0.04) * 0.5 = -0.01$，因此可以猜的次數的期望值為 $1.2/0.01 = 120$，要拿到 128 個 output 是有機會的。

2. Single

Concept

這題使用了 Elliptic curve 加密。
一開始先試著用 curve 上的兩個點找出參數 a, b。
接著會發現利用算出來的 a, b 滿足 $4 * a^3 + 27 * b^2 = 0 \pmod{p}$
因此可以得知這條 curve 是一個 singular curve。

接著透過 singular 的性質，去計算這個 curve 的 root，發現他有二重根。因此可以透過講義上 Singular Curve Node 的 homomorphism 性質: $\varphi(P+Q) = \varphi(P) + \varphi(Q)$, Define $\varphi(P(x,y)) = \frac{y+\sqrt{\alpha-\beta}(x-\alpha)}{y-\sqrt{\alpha-\beta}(x-\alpha)}$，將問題簡化為 DLP on (F_p, x)，$\varphi(dP) = \varphi(P)^d$

接著，試著去質因數分解 $order - 1$，發現可以分解成較小的質數群。
因此可以將簡化後的問題使用 Pohilme hellman 演算法去算出答案
簡化後的問題: $\varphi(dP) = \varphi(P)^d$
Note: 由於 flag 是用 dA 加密的，因此這邊的 P 點用 A 來代入。

最後成功解出 dA 後，便可以重現密鑰 k，再透過與密文 enc 進行 xor 運算成功得到 Flag

3. HNP-revenge

Concept

關鍵在這行

```
k = int(md5(b'secret').hexdigest() + md5(long_to_bytes(prikey.secret_multiplier)
+ h).hexdigest(), 16)
```

透過這行可以得知密鑰 k 的前 128bit 是固定的，後 128bit 會隨著使用者輸入而改變。因此可以嘗試使 Lattice 進行 known high bit attack。

首先我們假設:

- $k_1 = s + k_1'$
- $k_2 = s + k_2'$

s 是已知的固定前 128bit 部分，k_1' 以及 k_2'(128 bit) 是我們想要解的部分，可以利用其數字較小的性質轉成 Lattice 的問題去解

接著透過 ECDSA 的公式可以推導出:
$k_1 + t * k_2 + u = 0 \pmod{n}$

- $t = s_1^{-1} * s_2 * r_1 * r_2^{-1}$
- $u = s_1^{-1} * r_1 * h_2 * r_2^{-1} - s_1^{-1} * h_1$

有了上述條件後，可以建立 Lattice 的基礎矩陣

$$B = \begin{pmatrix} n & 0 & 0 \\ t & 1 & 0 \\ (u+s) & -s & K \end{pmatrix}$$

透過此矩陣可以得知 vector $(-k_1', k_2', K)$ 在此 lattice 裏面，因為 $(-q, k_2, 1) * B = (-q * n + k_2 * t + (u+s), k_2 - s, K) = (-k_1', k_2', K)$ 。Note: q 為某個整數

由以上可假設 k_1', k_2' 皆小於 K，且 K 須滿足 $K < det(B)^{\frac{1}{3}} = (n * K)^{\frac{1}{3}} => K < n^{\frac{1}{2}}$
k_1', k_2' 皆小於 $n^{\frac{1}{2}} = 2^{128}$，因此可以透過解此 Lattice 來得到 vector $(-k_1', k_2', K)$

接著透過 LLL 演算法運算便可以得到 k_1'，進而推導出 d。
而透過 d 可以得知 private key，並且可以用來在第三次的對話中幫"Kuruwa" 字串進行簽章，傳送簽章並成功拿到 Flag。