破密學 Final CTF Report

B01901169 王愷

Github: https://github.com/b01901169/crypto/tree/master/final/solution

RSA 160, 256

我一開始使用 sage 內建的函數 factor(n)，可以用來分解比較小的質數(例如 RSA 160)。不過後來再嘗試做 RSA 256 的時候發現複雜度似乎太大，因此無法如期分解完，因此改採用 cado-nfs () 這個人家寫好的 tools，可以使用 number field sieve 來對整數做分解，按照它裡面所寫的安裝完了之後，進到資料夾裡面，執行

./cado-nfs.py 90377629292003121684002147101760858109247336549001090677693 -t 4

其中後面的 –t 為指定使用多少個 cpu 來運算，這個工具可以很快的分解出 RSA 256 (約莫數分鐘內)，因此我就使用他來解出 RSA 256 的題目。

至於 RSA 512，即使使用了 cado-nfs，所需要花的時間還是太大，短時間內無法分解。

ECC 60, 100

```
P = 1136909012913895610619299
A = 1049529962351829133028442
B = 41792513741385409236432 3
G = (947568670657555140669246, 97108866468899398617789)
kG= (926978017941551607526856, 916111890075709280798539)

E = EllipticCurve(Integers(P),[A,B])
GG = E(G[0],G[1])
kGG = E(kG[0],kG[1])

k = GG.discrete_log(kGG)
print k*GG
print k
```

ECC 我也是使用 sage 內建的 discrete_log function 來解，將題目給的參數都設好之後直接執行 sage ecc_solution.sage，如果順利的話，就會 print 出 discrete log 的解。

在 ECC 60 的部分都可以輕鬆解出來，不用花 1 分鐘就可以順利解出 discrete log problem。

而在 ECC 80 的部分，嘗試過所有剩下來的題目，都無法順利在可以接受的時間內解完，複雜度太大因此並沒有解出 ECC 80 的部分。

ECC 100 則是有些題目出得並沒有很複雜，sage 可以在可接受的時間內解出 discrete log，估計是因為 Pollard ρ method 的循環節恰巧在這些題目之中比較短，因此很快可以找到碰撞並分解出 discrete log。