Introduction to Fintech Homework 7

B05102074 何青儒

Use the elliptic curve “secp256k1” as Bitcoin and Ethereum. Let G be the base point in the standard. Let d be the last 6 digits of your student ID number.

Thus, d = 102074

```
# private key
d = 102074

# base point G's coordinate of secp256k1
# Ref : http://www.secg.org/sec2-v2.pdf
# 其中 0x 代表十六進位
G_x = 0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798
G_y = 0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8

# finite field F_p
p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F

# order n
n = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
```

生成要用的橢圓曲線，並檢查n*G：

```
# 生成一個群，大小是p
K = GF(p)

# 生成一條曲線
E = EllipticCurve(K,[0,0,0,0,7])
print "E is an", E

# base point in the standard
G = E(G_x,G_y)

print "Base Point G is:", G
print "n*G = ",n*G
```

```
> E is an Elliptic Curve defined by y^2 = x^3 + 7 over Finite Field of size 115792089237316195423570985008687907853269984665640564039457584007908834671663
> Based point G is: (55066263022277343669578718895168534326250603453777594175500187360389116729240 : 32670510020758816978083085130507043184471273380659243275938904335757337482424 : 1)
> n*G = (0 : 1 : 0)
```

1. Evaluate 4G.

```
print "4G is :", 4*G
```

```
> 4G is (10338857399563508035974916425421659830878883530402360147780309523428649499368 3 : 37057141145242123013015316630864329550140216928701153669873286428255828810018 : 1)
```

> 4G is
>
> 10338857399563508035974916425421659830878883530402360147780309523428649499368 3,
>
> 37057141145242123013015316630864329550140216928701153669873286428255828810018

2. Evaluate 5G.

```
print "5G is :", 5*G
```

```
> 5G is (21505829891763648114329055987619236494102133314575206970830385799158076338148 :
98003708678762621233683240503080860129026887322874138805529884920309963580118 : 1)
```

> 5G is
>
> 21505829891763648114329055987619236494102133314575206970830385799158076338148,
>
> 98003708678762621233683240503080860129026887322874138805529884920309963580118

3. Evaluate Q = dG.

```
Q = d*G
print "Q is :", Q
print "n*Q = ", n*Q
```

```
> Q is (72566659804716573580475970416012857107125943565472618733613624180150592025225 :
54689446541871085750192196699997313834594274267540733146976629497494349135969 : 1)
> n*Q: (0 : 1 : 0)
```

> Q is
>
> 72566659804716573580475970416012857107125943565472618733613624180150592025225,
>
> 54689446541871085750192196699997313834594274267540733146976629497494349135969

4. With standard Double-and Add algorithm for scalar multiplications, how many doubles and additions respectively are required to evaluate dG?

$$d = 102074 = 2^{16} + 2^{15} + 2^{11} + 2^{10} + 2^9 + 2^7 + 2^5 + 2^4 + 2^3 + 2^1$$

> 所以要做至少 16 次的 double 跟 9 次的 addition。

5. Note that it is effortless to find -P from any P on a curve. If the addition of an inverse point is allowed, try your best to evaluate dG as fast as possible. Hint: 31P = 2(2(2(2P))) -P

$$d = 102074 = 2^{17} - 2^{14} - 2^{13} - 2^{12} - 2^8 - 2^6 - 2^2 - 2^1$$

> 所以要做至少 17 次的 double 跟 8 次的 addition。

6. Take a Bitcoin transaction as you wish. Sign the transaction with a random number k and your private key d.

- 先計算出公鑰。

```
# k is a random integer
# z set by self
k = ZZ.random_element(n)
z = 10
print "The random integer k is:",k

# 公鑰
public_key = k*G
print "PUBLIC KEY(x,y)", public_key
```

```
> The random integer k is: 860860796002914378609955987518909030864144233580831568889527798912649046551932
> PUBLIC KEY(x,y) (736419613793812452419017918508399418548123596907819350044636617744620558659240 :
6860112931907330433702225171120693841164196502702675967284602458367355615843O : 1)
```

- 再算出簽名對 (r,s)。

```
r = public_key[0] % n
s = (z + r*d)/k % n
print "(r,s) : (",r,",",s,")"
```

```
> (r,s) : ( 736419613793812452419017918508399418548123596907819350044636617744620558659240 ,
8232453627295413308141283881862812043541953990365571745101308204320279822290l )
```

7. Verify the digital signature with your public key Q.

- 先求出 w , u , v 。

```
w = 1/s % n
print "w = 1/s mod n:", w

u = z * w % n
print "u = z*w mod n:",u

v = r * w % n
print "v = r*w mod n:",v
```

```
> w = 1/s mod n: 106661726896524458302235743793986733173986732672844214395205151260585707540197
> u = z*w mod n: 244884658293988242102185728616761610643292482167680045086050443321936219529370
> v = r*w mod n: 413673993743925316430196751019133517105589739226040595878329469488230141694770
```

- 再算出新的點 F = uG + vQ。

```
F = (u*G) + (v*Q)
print "F:", F
```

```
F: (736419613793812452419017918508399418548123596907819350044636617744620558659240 :
6860112931907330433702225171120693841164196502702675967284602458367355615843O : 1)
```

- 檢查 F 這點的 x 座標是否符合簽章規則。

```

valid_x = F[0]

print "檢查是否符合簽章：", valid_x % n
print "和原本的 r 做比較：", r % n
print valid_x % n == r % n
```

```
> 檢查是否符合簽章： 736419613793812452419017918508399418548123596907819350044636617744620558659 24
> 和原本的 r 做比較： 736419613793812452419017918508399418548123596907819350044636617744620558659 24
> True
```