FinTech HW7

歐政鷹 R07922142 資工所碩一

secp256k1's elliptic curve:

$$y^2 = x^3 + 7$$

$\therefore a = 0\,, b = 7$

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

$p = 2^{256} - 2^{32} - 2^{9} - 2^{8} - 2^{7} - 2^{6} - 2^{4} - 1$

Base point G

= (0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8)

1. Evaluate 4G.

```
# secp256k1's elliptic curve y^2 = x^3  + 7
p = 2**256 - 2**32 - 2**9 - 2**8 - 2**7 - 2**6 - 2**4 - 1
a = 0
b = 7
print "Is P a prime : " , p.is_prime()

GX = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
GY = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

ec = EllipticCurve( GF(p) ,  [a,b] )
G = ec( GX , GY )
print "Is G a basepoint :" , G.order() == ec.order()
print

print "4G :" , 4*G  # Q1
#print "5G :" , 5*G  # Q2
```

```
Is P a prime :  True
Is G a basepoint : True

4G : (103388573995635080359749164254216598308788835304023601477803095234286494993683 :
37057141145242123013015316630864329550140216928701153669873286428255828810018 : 1)
```

2. Evaluate 5G.

```
# secp256k1's elliptic curve y^2 = x^3  + 7
p = 2**256 - 2**32 - 2**9 - 2**8 - 2**7 - 2**6 - 2**4 - 1
a = 0
b = 7
print "Is P a prime : " , p.is_prime()

GX = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
GY = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

ec = EllipticCurve( GF(p) ,  [a,b] )
G = ec( GX , GY )
print "Is G a basepoint :" , G.order() == ec.order()
print

#print "4G :" , 4*G  # Q1
print "5G :" , 5*G  # Q2
```

```
Is P a prime :  True
Is G a basepoint : True

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

3. Evaluate Q = dG.

```
# secp256k1's elliptic curve y^2 = x^3  + 7
p = 2**256 - 2**32 - 2**9 - 2**8 - 2**7 - 2**6 - 2**4 - 1
a = 0
b = 7
print "Is P a prime : " , p.is_prime()

GX = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
GY = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

ec = EllipticCurve( GF(p) ,  [a,b] )
G = ec( GX , GY )
print "Is G a basepoint :" , G.order() == ec.order()
print

#print "4G :" , 4*G  # Q1
#print "5G :" , 5*G  # Q2

d = 922142
print "Q = dG =" , d*G  # Q3
```

```
Is P a prime :  True
Is G a basepoint : True

Q = dG = (69133556303555002143213736136381315039087403870379930500584747802545114065746 :
97729634741569080413487417347456250037193562353438981450223859088234873450447 : 1)
```

4. Double-and Add algorithm.

```
d = (922142)10 = (11100001001000011110)2

i        di      |        a                        b
--       --      |        --                       --
1        1       |        1P+1P=2P                 2P+1P=3P
2        1       |        3P+3P=6P                 6P+1P=7P
3        0       |        7P+7P=14P
4        0       |        14P+14P=28P
5        0       |        28P+28P=56P
6        0       |        56P+56P=112P
7        1       |        112P+112P=224P           224P+1P=225P
8        0       |        225P+225P=450P
9        0       |        450P+450P=900P
10       1       |        900P+900P=1800P          1800P+1P=1801P
11       0       |        1801P+1801P=3602P
12       0       |        3602P+3602P=7204P
13       0       |        7204P+7204P=14408P
14       0       |        14408P+14408P=28816P
15       1       |        28816P+28816P=57632P     57632P+1P=57633P
16       1       |        57633P+57633P=115266P    115266P+1P=115267P
17       1       |        115267P+115267P=230534P  230534P+1P=230535P
18       1       |        230535P+230535P=461070P  461070P+1P=461071P
19       0       |        461071P+461071P=922142P

Q=dP is calculated in 27th step
```

$\therefore \text{step} = 27$

5. $\text{d} = (922142)_{10} = (1110\ 0001\ 0010\ 0001\ 1110)_2$

從 2 進制角度來看，最後$(01\ 1110)_2$中執行了 4 次加法，若改寫成$(10\ 0000 - 10)_2$ 可改成只利用一次加法及減法來運算，減少運算次數。

let $\delta = (1110\ 0001\ 0010\ 0010\ 0000)_2 = (922144)_{10}$

```
d = (922144)10 = (11100001001000100000)2

i        di      |        a                        b
--       --      |        --                       --
1        1       |        1P+1P=2P                 2P+1P=3P
2        1       |        3P+3P=6P                 6P+1P=7P
3        0       |        7P+7P=14P
4        0       |        14P+14P=28P
5        0       |        28P+28P=56P
6        0       |        56P+56P=112P
7        1       |        112P+112P=224P           224P+1P=225P
8        0       |        225P+225P=450P
9        0       |        450P+450P=900P
10       1       |        900P+900P=1800P          1800P+1P=1801P
11       0       |        1801P+1801P=3602P
12       0       |        3602P+3602P=7204P
13       0       |        7204P+7204P=14408P
14       1       |        14408P+14408P=28816P     28816P+1P=28817P
15       0       |        28817P+28817P=57634P
16       0       |        57634P+57634P=115268P
17       0       |        115268P+115268P=230536P
18       0       |        230536P+230536P=461072P
19       0       |        461072P+461072P=922144P

Q=dP is calculated in 24th step
```

$$\therefore d = \delta - (10)_2 = (1110\ 0001\ 0010\ 001\ 1110)_2 = (922142)_{10}$$

$$\therefore \text{step} = 24 + 1 = 25$$

6. Sign the transaction with a random number k and your private key d.

```
# secp256k1's elliptic curve y^2 = x^3  + 7
p = 2**256 - 2**32 - 2**9 - 2**8 - 2**7 - 2**6 - 2**4 - 1
a = 0
b = 7
print "Is P a prime : " , p.is_prime()

GX = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
GY = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

ec = EllipticCurve( GF(p) ,  [a,b] )
G = ec( GX , GY )
print "Is G a basepoint :" , G.order() == ec.order()
print

#print "4G :" , 4*G  # Q1
#print "5G :" , 5*G  # Q2

d = 922142
#print "Q = dG =" , d*G  # Q3

#Q6
private_key = d
pubile_key = d*G
n = G.order()
k = ZZ.random_element(1,n-1)
k_inverse = k.inverse_mod(n)
z = ZZ.random_element(1,2**256-1)

curve_x , curve_y, space = k*G
curve_x , curve_y = (int(curve_x), int(curve_y))
print 'Curve point :' , curve_x ,curve_y , '\n'

r = curve_x % n
s = (k_inverse * (z + r*d)) % n
print 'The Signature is the pair :' , (r,s)
print
```

```
Is P a prime :  True
Is G a basepoint : True

Curve point : 70228438663104990622978102788092995139187590747290562484975438226038745116822
39805993042130232739016653224653643950313516716355456054801269141893294610818

The Signature is the pair : (70228438663104990622978102788092995139187590747290562484975438226038745116822,
43028073257605019605740744457182318075964782580929013481888568887394930548174)
```

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

```
# secp256k1's elliptic curve y^2 = x^3  + 7
p = 2**256 - 2**32 - 2**9 - 2**8 - 2**7 - 2**6 - 2**4 - 1
a = 0
b = 7
print "Is P a prime : " , p.is_prime()

GX = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
GY = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

ec = EllipticCurve( GF(p) ,  [a,b] )
G = ec( GX , GY )
print "Is G a basepoint :" , G.order() == ec.order()
print

#print "4G :" , 4*G  # Q1
#print "5G :" , 5*G  # Q2

d = 922142
#print "Q = dG =" , d*G  # Q3

#Q6
private_key = d
pubile_key = d*G
n = G.order()
k = ZZ.random_element(1,n-1)
k_inverse = k.inverse_mod(n)
z = ZZ.random_element(1,2**256-1)

curve_x , curve_y, space = k*G
curve_x , curve_y = (int(curve_x), int(curve_y))
print 'Curve point :' , curve_x ,curve_y , '\n'

r = curve_x % n
s = (k_inverse * (z + r*d)) % n
print 'The Signature is the pair :' , (r,s)
print
#Q7
O = n*G
Q = pubile_key
print 'check pubile key is not equal to the identity element O :' , Q != O
print 'check pubile key lies on the curve :' , ec(Q[0],Q[1]) is not None
print 'check n*Q = O :' , n*Q == O

w = s.inverse_mod(n)
u_1 = (z*w) % n
u_2 = (r*w) % n
curve_x , curve_y, space = u_1*G + u_2*Q
curve_x , curve_y = (int(curve_x), int(curve_y))
print "Is signature valid :" , (r%n) == (curve_x%n)
```

```
Is P a prime :  True
Is G a basepoint : True

Curve point : 70228438663104990622978102788092995139187590747290562484975438226038745116822
39805993042130232739016653224653643950313516716355456054801269141893294610818

The Signature is the pair : (70228438663104990622978102788092995139187590747290562484975438226038745116822,
43028073257605019605740744457182318075964782580929013481888568887394930548174)

check pubile key is not equal to the identitly element O : True
check pubile key lies on the curve : True
check n*Q = O : True
Is signature valid : True
```