Fintech hw6

R08944052 斯曉宇

Private key d = 944052

1. 4G =

(10338857399563508035974916425421659830878883530402360147780309523428649499368 3,

37057141145242123013015316630864329550140216928701153669873286428255828810018)

2. 5G =

(21505829891763648114329055987619236494102133314575206970830385799158076338148 ,

98003708678762621233683240503080860129026887322874138805529884920309963580118)

3. Q = dG = 944052G

(69911091709759721880789243265636197136441885614673903871710885378333602519135,

37845052702926721736303725680106054418295406492842005363183630332713819274285)

4. Q = dG = 944052G = $(11100110011110110100)_2$G

step	binary	operation	value		step	binary	operation	value
0	1	Initialization	G		10	1	Double and add	1843G
1	1	Double and add	3G		11	1	Double and add	3687G
2	1	Double and add	7G		12	1	Double and add	7375G
3	0	Double	14G		13	0	Double	14750G
4	0	Double	28G		14	1	Double and add	29501G
5	1	Double and add	57G		15	1	Double and add	59003G
6	1	Double and add	115G		16	0	Double	118006G
7	0	Double	230G		17	1	Double and add	236013G
8	0	Double	460G		18	0	Double	472026G
9	1	Double and add	921G		19	0	Double	944052G

Doubles = 19

Adds = 11

5. 從第二位開始左至右觀察二進制位

d = 944052 = $(11100110011110110100)_2$

Init 1

Add 1+1 = 10，double 10*10 =100 ，double 100*10 =1000

Add inverse 1000-1 = 111，double 111*10=1110

依此類推得到下面的演算法

1）如果出現連續 n 個 1 後面跟著一個 0，where n≥2。遇到 1 的部

分則可以先做 add，然後 double n 次，再 add an inverse point。

2）如果只有一個 1 後面跟著一個 0，遇到 1 的部分則做 double and add。

3）遇到 0，直接做 double。

轉換成程式碼（sage）如下：

```
#bin_d = "11100110011110110100"
d = 944052
bin_d = d.binary()
ones = 0
op = ["init"]
for d in bin_d[1:]:
      if d == "1":
            ones += 1
      else:
            if ones == 1:
                  op += ["double"] + ["add"]
            elif ones >= 2:
                  op += ["add"] + ones*["double"] + ["inverse"]
            op += ["double"]
            ones = 0
print "operations = ",op
```

```
operations =  ['init', 'add', 'double', 'double', 'inverse', 'double', 'double',
'add', 'double', 'double', 'inverse', 'double', 'double', 'add', 'double',
'double', 'double', 'double', 'inverse', 'double', 'add', 'double', 'double',
'inverse', 'double', 'double', 'add', 'double', 'double']
```

```
# check if operations are right
# translate operations to d
def reverse_check(operation):
      d = 1
      add = double = inverse = 0
      for op in operation[1:]:
            if op == "add":
                  d += 1
                  add += 1
            elif op == "double":
                  d *= 2
                  double += 1
            elif op == "inverse":
                  d -= 1
                  inverse += 1
      print "doubles=",double,"\nadds=",add, "\nadd inverse=",inverse,"\nd is",d

print "private key d in decimal is 944052"
print "Checking translated operation:\n", reverse_check(op)
```

```
private key d in decimal is 944052
Checking translated operation:
doubles= 19
adds= 5
add inverse= 4
d is 944052
```

New Doubles = 19

New Adds = 5

Add inverse = 4

6.

d = 944052 #private key

Sign a transaction

z=ff0d5adde54293a9e41d7e2011244ac1cf03d6200a0c145bd8025ee2613

da21b

k = 1024

```
#橢圓曲線——有限體
F = GF(2^256 - 2^32 - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1)

#橢圓曲線
a = 0
b = 7
EC = EllipticCurve(F, [a, b])

Gx = 0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798
Gy = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
G = EC(Gx, Gy)
n = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

d = 944052 #private key
Q = d*G   #public key
e = 0xff0d5adde54293a9e41d7e2011244ac1cf03d6200a0c145bd8025ee2613da21b
e_bin = e.binary() #轉成二進位
n_bin = n.binary()
len(n_bin)==len(e_bin) #True --> L_e = L_n
```

```
True
```

```
# Sign a transaction
z = 0xff0d5adde54293a9e41d7e2011244ac1cf03d6200a0c145bd8025ee2613da21b
k = 1024
kG = k*G
print "(x1,y1)=",kG
```

```
(x1,y1)=
(16339661702852967382840638396154683021742565846854739320600712521008743256863 :
36728284022334234592863792440353097018527397507662959793213172092233334129261 : 1)
```

```
#x1 = 16339661702852967382840638396154683021742565846854739320600712521008743256863
x1 = int(kG[0])
r = x1 % n
s = (k.inverse_mod(n) * (z + r * d)) % n
print "r=",r
print "s=",s
print "signature is",(r,s)
```

```
r= 16339661702852967382840638396154683021742565846854739320600712521008743256863
s= 66863891821311871813551683259657101887910968366794819008612068819912310191 73
signature is
(16339661702852967382840638396154683021742565846854739320600712521008743256863,
66863891821311871813551683259657101887910968366794819008612068819912310191 73)
```

7. Verification

```
#Verification
w = s.inverse_mod(n) % n
u1 = z*w % n
u2 = r*w % n
verified_xy = u1 * G + u2 * Q
print "Verified (x1,y1)=",verified_xy
print "r==verified x?",verified_xy[0]==r
if verified_xy[0]==r:
      print "Transaction is valid"
```

```
Verified (x1,y1)=
(16339661702852967382840638396154683021742565846854739320600712521008743256863 :
36728284022334234592863792440353097018527397507662959793213172092233334129261 : 1)
r==verified x? True
Transaction is valid
```