# Introduction to FinTech
# Assignment for Bitcoin/Blockchain

M11215032 葉品和

Use the elliptic curve “secp256k1” as Bitcoin and Ethereum. Let G be the base point in the standard. Let d be the last 4 digits of your student ID number.

**Parameters**

| Name | Value |
|---|---|
| p | 0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f |
| a | 0x0000000000000000000000000000000000000000000000000000000000000000 |
| b | 0x0000000000000000000000000000000000000000000000000000000000000007 |
| G | (0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798, 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8) |
| n | 0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141 |
| h | 0x1 |

**d = 5032**

```
G =  (55066263022277343669578718895168534326250603453777594175500187360389116729240 : 32670510020758816978083085130507043184471273380659243275938904335757337482424 : 1)

1. 4G =  (103388573995635080359749164254216598308788835304023601477803095234286494993683 : 37057141145242123013015316630864329550140216928701153669873286428255828810018 : 1)
2. 5G =  (21505829891763648114329055987619236494102133314575206970830385799158076338148 : 98003708678762621233683240503080860129026887322874138805529884920309963580118 : 1)
3. Q = d*G =  (75203615339765883360117800256938721265645905885881269054284209794029431069746 : 85406770260463487610787142836915240400243566110756911651070945058173682096418 : 1)
4. d = 5032, Doubles =  12 , Additions =  5
6. signature (r, s) = ( 101288984487691882293336356097393489236813260009637475882017091422520890252670 ,  778740184682468695550244997269632875206237944911803001035844365798621104141520 )
7. Verification = True
(a, b, c)
8. p(1) = 10, p(2) = 20, and p(3) = d:
[[a == 2501, b == -7493, c == 5002]]
```

1. Evaluate 4G.

```
4G =
(10338857399563508035974916425421659830878883530402360147780309523428649499368
3 :
37057141145242123013015316630864329550140216928701153669873286428255828810018 : 1)
```

2. Evaluate 5G.

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

3. Evaluate Q = dG.

Q = d*G =
(75203615339765883360117800256938721265645905885881269054284209794029431069746 :
85406770260463487610787142836915240400243566110756911651070945058173682096418 : 1)

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

d = 5032, Doubles = 12 , Additions = 5

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(2(2P)))) −P.

5032 的二進位為 1001110101000，其中「0」的數量多於「1」的數量，也沒有大量連續的「1」，基本上正常做 Doubles = 12 與 Additions = 5 應該會是最少的步驟。

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

```
Fn = FiniteField(n)

def hashit(msg):
    return Integer('0x' + hashlib.sha256(msg.encode()).hexdigest())

def ecdsa_sign(d, m):
    r = 0
    s = 0
    while s == 0:
        k = 1
        while r == 0:
            k = randint(1, n - 1)
            Q = k * G
            (x1, y1) = Q.xy()
            r = Fn(x1)
        e = hashit(m)
        s = Fn(k) ^ (-1) * (e + d * r)
    return [r, s]
```

先在 1 ~ n-1 的範圍內找出隨機整數 k，接著計算 curve point = k * G，計算 $r = x_1 \bmod(n)$，若 r = 0 則重新找另一個隨機整數 k，繼續計算 $s = k^{-1} * (z + rd_A) \bmod(n)$，若 s = 0則同樣重新找 k，若 r 與 s 都不為 0，完成簽章(r, s)。

```
signature (r, s) = (
691588406876213119993188366684262038704888794611858368171677748704178269861 2 ,
101567484305318658097682723221997115577217861347847613526936742786249647757220 )
```

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

```
def ecdsa_verify(Q, m, r, s):
    e = hashit(m)
    w = s ^ (-1)
    u1 = (e * w)
    u2 = (r * w)
    P1 = Integer(u1) * G
    P2 = Integer(u2) * Q
    X = P1 + P2
    (x, y) = X.xy()
    v = Fn(x)
    return v == r
```

將 message 經過 HASH 得到 e，先計算 $w = s^{-1} \bmod(n)$，接著代入 $u_1 = e * w \bmod(n)$ 與 $u_2 = r * w \bmod(n)$，計算 curve point $(x1, y1) = u_1 * G + u_2 * Q$，若 $r = x_1$ 則驗章成功。

```
Verification = True
```

8. Over $Z_{10007}$, construct the quadratic polynomial p(x) with p(1) = 10, p(2) = 20, and p(3) = d

```
var('a b c')
eq1 = a + b + c == 10
eq2 = 4*a + 2*b + c == 20
eq3 = 9*a + 3*b + c == 5032
print("8. p(1) = 10, p(2) = 20, and p(3) = d: ")
solve([eq1, eq2, eq3], a, b, c)
```

```
[a == 2501, b == -7493, c == 5002]
p(x) = 2501*x^2 - 7493 * x + 5002
```