第6章：橢圓曲線密碼系統

6-1 橢圓曲線密碼系統(ECC)簡介

6-2 橢圓曲線點乘法

6-3 橢圓曲線數位金鑰交換演算法ECDH

6-4 橢圓曲線數位簽章演算法

6-5 橢圓曲線參數

橢圓曲線密碼系統(ECC)

- 橢圓曲線密碼系統（Elliptic Curve Cryptosystem， ECC）
- 1985年由Koblitz與Miller各別提出的公開金鑰密碼學技術
- 國際標準與國家標準如ISO 11770-3、ANSI X9.62、IEEE P1363 、FIPS 186-2 、 GB15629.11-2003 (WAPI, Wireless Authentication Privacy Infrastructure) 等
- 橢圓曲線的技術不只能應用在密碼學加解密、數位簽章、金鑰交換等，也能應用於大數分解(factorization)與質數判斷(primality testing)
- 在相同的安全強度下，ECC的密碼學金鑰長度可遠較其他公開金鑰密碼系統(如RSA)小且處理速度較快，意即ECC每個金鑰位元所能提供的安全性遠超過其他公開金鑰密碼系統，這使得ECC非常適合利用於如智慧卡或手機無線行動裝置等記憶體有限的環境中

相同安全性時RSA與ECC金鑰長度比較

安全性 / 演算法	2^{80}	2^{112}	2^{128}	2^{192}	2^{256}
RSA長度(位元)	1024	2048	3072	7680	15360
ECC長度(位元)	160	224	256	384	512
RSA:ECC金鑰長度比	6:1	9:1	12:1	20:1	30:1

[NIST]

橢圓曲線

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

- 滿足上述方程式的所有點(x, y)及一個無限遠點(point at infinity)∞所形成的集合，其中座標x與y屬於某個有限體(finite field)
- 橢圓曲線的級數(order)為曲線上包含無限遠點的所有點的數目
- 有限體為質數體(prime field， GF(p))、二元體(binary field ，GF(2^n))、最佳擴展體(optimal extension field ，GF (p^n))等三種
- Hasse定理：如果採用有限體GF(q)則橢圓曲線的級數滿足

$$q+1-2\sqrt{q} \le order \le q+1+2\sqrt{q}$$

橢圓曲線範例一

- 質數體為GF(5)且橢圓曲線公式

  $y^2 = x^3 + x + 1$
- 這個橢圓曲線上的點，除無限遠點∞外，另有8個點：(0,1),(0,4),(2,1),(2,4),(3,1),(3,4),(4,3),(4,2)點的座標值屬於GF(5) 。
- 因為共有9點，所以此曲線的級數(order)為9

橢圓曲線範例二

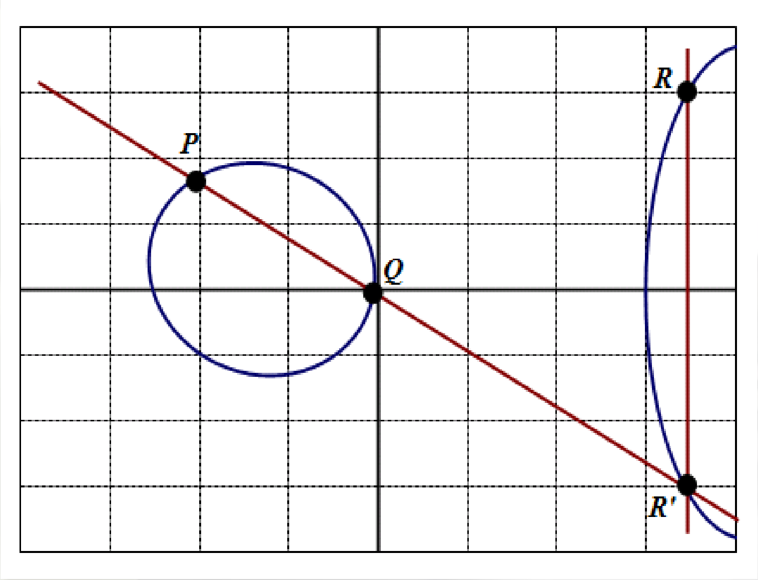
- 質數體為GF(11)且橢圓曲線公式

  $y^2 = x^3 + x + 1$

- 這個橢圓曲線上的點，除無限遠點∞外，另有13個點：
  (0,1),(0,10),(1,6),(1,5),(2,0),(3,8),(3,3),(4,6),(4,5),(6,6),(6,5),(8,2),(8,9) 點的座標值屬於GF(11)
- 因為共有14點，所以此曲線的級數(order)為14

橢圓曲線加法

- 幾何上如果要計算相異兩點*P*與*Q*的和，則我們先找出通過這兩點的直線，然後找出這條直線與橢圓曲線相交的第三點，再將此點對*x*軸做鏡射得到和。如果橢圓曲線上的某兩點共線的話，兩點相加之和就是∞



橢圓曲線加法公式(質數體)

- 若$P = (x_1, y_1)$與$Q = (x_2, y_2)$為橢圓曲線上的任意兩點，但$P \neq \infty \neq Q$，且選取質數體(此時橢圓曲線方程式為$y^2 = x^3 + ax + b$)，則我們有下列兩點加法的運算規則：
- 1. $P + \infty = \infty + P = P$
- 2. $P + (-P) = (x_1, y_1) + (x_1, -y_1) = \infty$
- 3. $P + Q = (x_3, y_3)$,

$$x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1,$$

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{如果} P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{如果} P = Q \end{cases}$$

橢圓曲線加法公式(二元體)

- 若$P = (x_1, y_1)$與$Q = (x_2, y_2)$為橢圓曲線上的任意兩點，但$P \neq \infty \neq Q$，且選取二元體(此時橢圓曲線方程式為 $y^2 + xy = x^3 + ax^2 + b$)，則規則3：

$P + Q = (x_3, y_3)$,

$$x_3 = \begin{cases} \lambda^2 + \lambda + x_1 + x_2 + a & \text{如果} P \neq Q \\ \lambda^2 + \lambda + a & \text{如果} P = Q \end{cases}$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$

$$\lambda = \begin{cases} \dfrac{y_2 + y_1}{x_2 + x_1} & \text{如果} P \neq Q \\ x_1 + \dfrac{x_1}{y_1} & \text{如果} P = Q \end{cases}$$

橢圓曲線點的級數

- 加法公式的計算(加法、減法、乘法、除法/反元素)須在相關的有限體進行，若選取質數體時僅需進行模算術(**modular arithmetic**)，若選取二元體則需進行多項式計算(**polynomial arithmetic**)
- 點乘法計算$k \bullet P$，其中k為正整數而P為橢圓曲線上的一個點
- 如果橢圓曲線上的一個點P我們找到最小的正整數n滿足$n \bullet P = \infty$ (無限遠點)，則n稱為點P的級數(**order**)
- 橢圓曲線上點的級數一定是曲線級數的因數

質數體橢圓曲線加法範例一

- 質數體為GF(5)且橢圓曲線公式$y^2 = x^3 + x + 1$
- G=(0,1), 2G= (4,2), 3G= G + 2G=(2,1), 4G=(3,4)，5G=(3,1)，6G=(2,4)，7G=(4,3)，8G=(0,4)，9G=∞。因為G =(0,1)時，9G=∞，所以此點G的級數為9
- 若我們選G=(2,1)，則2G=(2,4)，3G=∞。因為G=(2,1)時，3G=∞，所以此點G的級數為3

質數體橢圓曲線加法範例二

- 質數體為GF(11)且橢圓曲線公式$y^2 = x^3 + x + 1$
- 若我們選G=(0,1)，利用公式(6.5)計算，則我們有2G=(3,3)，3G=(6,6)， 4G=(6,5)， 5G=(3,8)，6G=(0,10)， 7G= ∞。因為G=(0,1)時，7G= ∞，所以此點G的級數為7

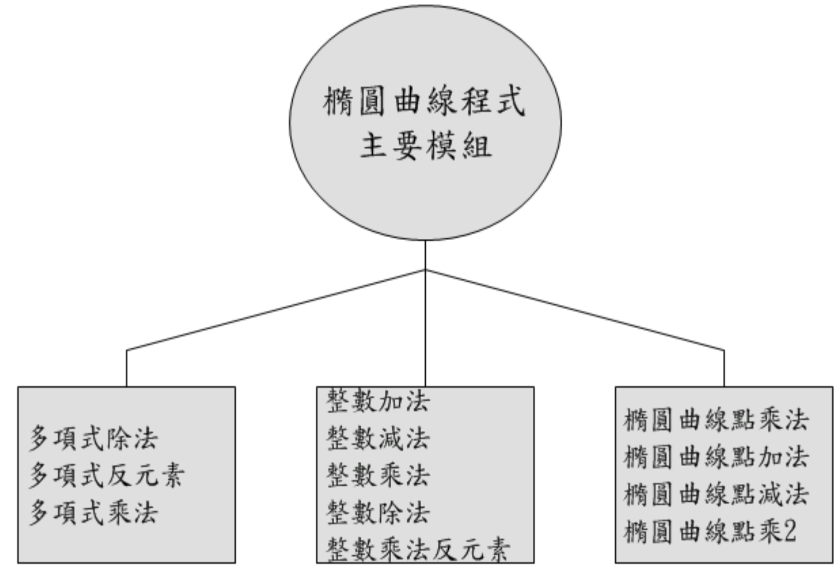
Mathematica

橢圓曲線加法的Mathematica程式：

```
If [Mod[x1 - x2, p] == 0,
    If [Mod[y1 + y2, p] == 0,
        slope = Mod[(3 x1^2 + g2) PowerMod[2 y1, -1, p], p]
    ],
    slope = Mod[(y2 - y1) PowerMod[x2 - x1, -1, p], p]
];
x3 =  Mod[slope^2 - x1 - x2, p];
y3 =  Mod[slope (x1-x3) - y1, p];
```

橢圓曲線密碼系統C語言函式庫主要模組

- 橢圓曲線密碼系統C語言函式庫模組及獨立可執行程式的主要模組包括三大部分：整數運算、多項式運算、橢圓曲線運算
- 如果採用質數體，則不需多項式運算，但是由於需要用到160位元以上的橢圓曲線以提供足夠的安全性，所以有限體的元素必須以某種資料結構(如陣列)來表示並進行運算，這也遠比第三章介紹的其他公開金鑰演算法來得複雜



橢圓曲線密碼系統的實現的考慮

- 有限體的選擇
- 橢圓曲線的挑選
- 有限體元素的運算(加法、減法、乘法、除法/反元素)
- 橢圓曲線點的運算(加法、減法、乘法)

由左向右二元演算法計算 $k \bullet P$ 點乘法

$$k \bullet P = \overbrace{P + P + \ldots + P}^{k\text{個點相加}}$$

$$k = \sum_{i=0}^{t-1} k_i 2^i$$

$k = (k_{t-1}\ k_{t-2}\ \ldots\ k_2\ k_1\ k_0)$，

$k_i \in \{0,1\}$ 且 $k_{t-1} \neq 0$

Step 1. $Q \leftarrow \infty$

Step 2. for i from t-1 downto 0 do

$Q \leftarrow Q + Q$ 相同點相加(point doubling)

If $k_i = 1$

then $Q \leftarrow Q + P$ 相異點相加(point addition)

如果係數k可用t位元的二進制數字來代表，通常其中平均有半數為1，半數為0。所以演算法需要t次相同點相加(point doubling)與約$t/2$的相異點相加(point addition)

由左向右二元演算法 $k \bullet P$ 範例

- **k = 12345 ，可用14位元的二進制數字來代表，12345 = (11000000111001)**

下表為 Step 2 執行後的結果。

i	Step 2 執行結束後
13	$Q=P$
12	$Q=3P$
11	$Q=6P$
10	$Q=12P$
9	$Q=24P$
8	$Q=48P$
7	$Q=96P$
6	$Q=192P$
5	$Q=385P$
4	$Q=771P$
3	$Q=1543P$
2	$Q=3086P$
1	$Q=6172P$
0	$Q=12345P$

總共只進行14次相同點相加與6次相異點相加，這當然遠比須進行k-1=12344次點相加有效率

由右向左二元演算法計算 $k \bullet P$ 點乘法

$$k \bullet P = \overbrace{P + P + \ldots + P}^{k\text{個點相加}}$$

$$k = \sum_{i=0}^{t-1} k_i 2^i$$

$k = (k_{t-1}\, k_{t-2} \ldots k_2\, k_1\, k_0)$，

$k_i \in \{0,1\}$ 且 $k_{t-1} \neq 0$

Step 1. $Q \leftarrow \infty$

Step 2. for i from 0 to t-1 do

If $k_i = 1$

then $Q \leftarrow Q + P$ 相異點相加(point addition)

$P \leftarrow P + P$ 相同點相加(point doubling)

- 點P的座標值會被改變

二元非相鄰形式(Non-Adjacent Form，NAF)

- 係數k的二元NAF很類似二進制數字，但每個位置除**0**與**1**外也允許**-1**，且不會有兩個相鄰非零的數字，$k = (k_{m-1}k_{m-2} \ldots k_2k_1k_0)$，$k_i \in \{-1,0,1\}$且$k_{m-1} \neq 0$，亦即

$$k = \sum_{i=0}^{m-1} k_i 2^i$$

- k的這個m位數字**NAF**是唯一且m最多比k的t位元二進制數字長度多**1**，即$m \leq t+1$，而且m個數字中平均只有**1/3**是非零

計算正整數*k*的二元NAF演算法

Step 1. $i \leftarrow 0$

Step 2. while $k \geq 1$ do

If k is odd

then $k_i \leftarrow 2 - (k \bmod 4)$, $k \leftarrow k - k_i$

else $k_i \leftarrow 0$

$k \leftarrow k/2$, $i \leftarrow i + 1$

二元NAF範例

- k = 12345
- 利用二元NAF演算法可算出k = (1 0 -1 0 0 0 0 0 1 0 0 -1 0 0 1)，每一位元代表的數值如下表第一列所示，第二列為k的一般二元表示法，第三列為k的NAF二元表示法

2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
0	1	1	0	0	0	0	0	0	1	1	1	0	0	1
1	0	-1	0	0	0	0	0	1	0	0	-1	0	0	1

由左向右二元NAF演算法計算$k \bullet P$

Step 1. $Q \leftarrow \infty$

Step 2. for $i \leftarrow t\text{-}1$ downto 0 do

$Q \leftarrow Q + Q$

If $k_i = 1$

then $Q \leftarrow Q + P$

If $k_i = -1$

then $Q \leftarrow Q - P$

- **Step 2**的執行過程是：先將相同點相加(point doubling)，接著如果$k_i = 1$則再進行相異點相加(point addition)，最後如果$k_i = -1$則再進行點相減(point subtraction)
- 由於$P - Q = P + (-Q)$，若$Q = (x_2, y_2)$則$-Q = (x_2, -y_2)$，所以質數體橢圓曲線的減法與加法工作量幾乎相同。由左向右二元NAF演算法只需要進行m次相同點相加與約$m/3$的點相加或相減，$m \approx t$，所以比二元演算法更有效率

由左向右二元NAF演算法範例

- *k* = 12345，二元NAF演算法執行，右表為Step 2執行後的結果
- 本範例中，總共進行15次相同點相加、3次相異點相加、與2次點相減

i	Step 2執行結束後
14	$Q=P$
13	$Q=2P$
12	$Q=3P$
11	$Q=6P$
10	$Q=12P$
9	$Q=24P$
8	$Q=48P$
7	$Q=96P$
6	$Q=193P$
5	$Q=386P$
4	$Q=772P$
3	$Q=1543P$
2	$Q=3086P$
1	$Q=6172P$
0	$Q=12345P$

橢圓曲線金鑰交換演算法ECDH

一、尋找雙方同意之系統橢圓曲線與基點G作為系統公開金鑰，且所有使用者均可用相同且公開的橢圓曲線參數與基點G，G的級數n要夠大(例如大於2^{160})

二、使用者A選擇祕密金鑰正整數x，$1 < x < n$，且計算$G_x = x \bullet G$，爾後G_x送給欲通信的使用者B。G_x為使用者A祕密金鑰x對應的公開金鑰

三、使用者B選擇祕密金鑰正整數y，其$1 < y < n$，且計算$G_y = y \bullet G$，爾後將G_y送給欲通信的使用者A。G_y為使用者B的祕密金鑰y對應的公開金鑰

四、使用者A利用自己的祕密金鑰整數x與收到的G_y，計算出公式

$$K_{AB} = x \bullet G_y = x \bullet (y \bullet G)$$

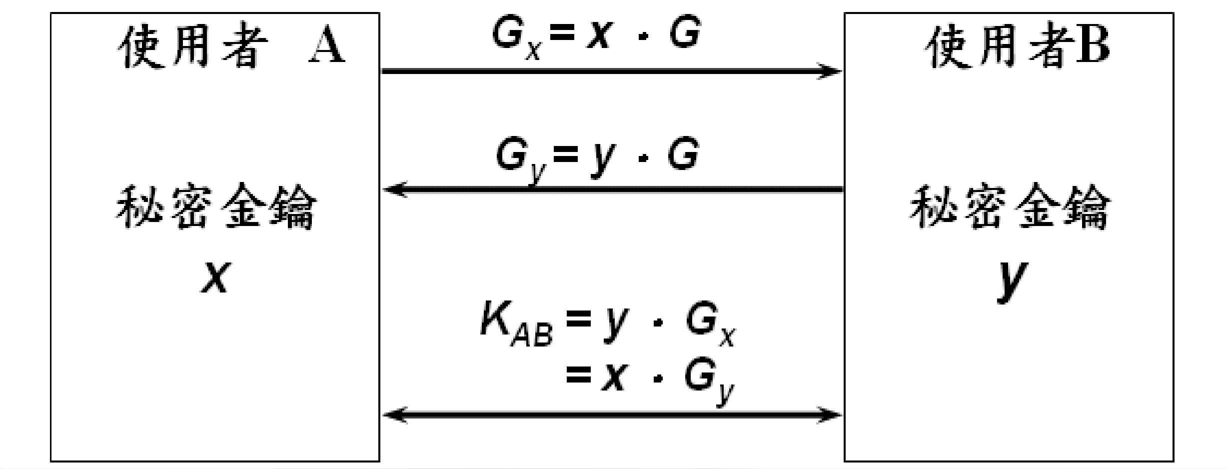
五、使用者B利用自己的祕密金鑰整數y與收到的G_x，計算出公式

$$K_{AB} = y \bullet G_x = y \bullet (x \bullet G)$$

六、雙方擁有共通的橢圓曲線點$(x \bullet y) \bullet G$。此點的x座標便可當作雙方的共通金鑰K_{AB}

ECDH (Elliptic curve Diffie-Hellman) 金鑰交換

- 如果網路中有窺視者想竊取資料，則只能取得系統公開的橢圓曲線參數與基點G以及通信過程中的兩筆資料$x \bullet G$與$y \bullet G$；欲從橢圓曲線基點G與$x \bullet G$ (或$y \bullet G$)尋找到x (或y)是很困難的，這就是所謂的橢圓曲線離散對數(elliptic-curve discrete logarithm)問題，其困難度與RSA的大數分解類似
- ECDH如同Diffie-Hellman公開金鑰交換演算法，本身無法抵擋惡意攻擊者在通信中間進行的主動式攔截攻擊，但可利用數位簽章技術，使通信雙方進行身分或金鑰資料認證，來防止中間人的攻擊



橢圓曲線數位簽章演算法(ECDSA)簽章產生

- 假設G是橢圓曲線系統基點且其級數為n，正整數d為簽署者的私密金鑰，而$Q = d \bullet G$則為簽署者的對應公開金鑰，$h(M)$為訊息M的雜湊函數值
- ECDSA對應於訊息M的簽章(r, s)產生步驟如下

Step 1 挑選一亂數k，$n - 1 \geq k \geq 1$

Step 2 計算 $k \bullet G = (x_1, y_1)$ 且 $r = x_1 \bmod n$.

如果 $r = 0$，則回到Step1

Step 3 計算$s = k^{-1} \{h(M) + d\, r\} \bmod n$

Step 4 如果$s = 0$，則回到Step 1。

ECDSA簽章檢驗

Step 1 計算$w = s^{-1} \bmod n$

Step 2 計算$u_1 = h(M)\, w \bmod n$ 與 $u_2 = r\, w \bmod n$

Step 3 計算$u_1 \bullet G + u_2 \bullet Q = (x_0, y_0)$ 與 $v = x_0 \bmod n$

Step 4 若且唯若$v = r$，則簽章正確。

ECDSA簽章驗證的理由

- 由於簽章產生Step 3有 $s \equiv k^{-1}\{h(M) + d\,r\} \bmod n$，所以$k \equiv s^{-1}\{h(M) + d\,r\} \bmod n$
- 利用簽章檢驗Step 1與Step 2可得$k \equiv w\,(h(M) + d\,r\}) \equiv u_1 + d\,u_2 \pmod{n}$
- 簽章檢驗Step 3有$(x_0, y_0) = u_1 \bullet G + u_2 \bullet Q = u_1 \bullet G + u_2 \bullet (d\ \ G) = (u_1 + d\,u_2) \bullet G = k \bullet G = (x_1, y_1)$
- 所以$r = x_1 \bmod n$與$v = x_0 \bmod n$兩數應該相等

數位簽章演算法RSA與ECDSA的比較

演算法	RSA	ECDSA
簽章長度	安全性2^{128}：384 位元組 安全性2^{192}：960位元組 安全性2^{256}：1920位元組	安全性2^{128}：64位元組(質數體) 安全性2^{192}：96位元組(質數體) 安全性2^{256}：132位元組(質數體)
安全基礎	大數分解。	橢圓曲線離散對數。
優點	演算法歷史悠久容易說明，且同時可用做加解密。	速度快，簽章長度小。
缺點	速度慢，簽章長度較大。	理論較難理解，且實作技術較複雜。

橢圓曲線參數

- 美國國家標準技術局(NIST)在FIPS 186-4標準文件裡推薦數組橢圓曲線參數
- 質數體有五種不同曲線
- $y^2 \equiv x^3 - 3x + b \pmod{p}$
- 係數a = -3，而這些曲線的級數n也為質數，基點(G_x, G_y)座標與係數b都已提供，且基點的級數$r = n$

NIST質數體橢圓曲線參數P-192

- p = 6277101735386680763835789423207666416083908700390324961279
- r = 6277101735386680763835789423176059013767194773182842284081
- b = (16進制) 64210519e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
- G_x = (16進制) 188da80eb03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
- G_y = (16進制) 07192b95ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811

NIST質數體橢圓曲線參數P-224

- p =
26959946667150639794667015087019630673557916260026308143510066298881
- r =
26959946667150639794667015087019625940457807714424391721682722368061
- b = b4050a85 0c04b3ab f5413256 5044b0b7 d7bfd8ba 270b3943 2355ffb4
- G_x = b70e0cbd 6bb4bf7f 321390b9 4a03c1d3 56c21122 343280d6 115c1d21
- G_y = bd376388 b5f723fb 4c22dfe6 cd4375a0 5a074764 44d58199 85007e34

NIST質數體橢圓曲線參數P-256

- p = 115792089210356248762697446949407573530086143415290314195533631308867097853951
- r = 115792089210356248762697446949407573529996955224135760342422259061068512044369
- b = 5ac635d8 aa3a93e7 b3ebbd55769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b
- G_x = 6b17d1f2 e12c4247 f8bce6e563a440f2 77037d81 2deb33a0 f4a13945 d898c296
- G_y = 4fe342e2 fe1a7f9b 8ee7eb4a7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5

NIST質數體橢圓曲線參數P-384

- p =
  39402006196394479212279040100143613805079739270465446667948293404245721771496870329047266088258938001861606973112319
- r =
  39402006196394479212279040100143613805079739270465446667946905279627659399113263569398956308152294913554433653942643
- b = b3312fa7 e23ee7e4 988e056b e3f82d19 181d9c6e fe814112 0314088f 5013875a c656398d 8a2ed19d 2a85c8ed d3ec2aef
- G_x = aa87ca22 be8b0537 8eb1c71e f320ad74 6e1d3b62 8ba79b98 59f741e082542a38 5502f25d bf55296c 3a545e38 72760ab7
- G_y = 3617de4a 96262c6f 5d9e98bf 9292dc29 f8f41dbd 289a147c e9da3113b5f0b8c0 0a60b1ce 1d7e819d 7a431d7c 90ea0e5f

NIST質數體橢圓曲線參數P-521

- p =
  6864797660130609714981900799081393217269435300143305409394463459
  1855431833976560521225596406614545549772963113914808580371219879997 16643812574028291115057151
- r = 686479766013060971498190079908139321726943530014330540939446345918554318339765539424505774633321719753296399637136332111386476 8612440380340372808892707005449
- b = 051 953eb961 8e1c9a1f 929a21a0 b68540ee a2da725b 99b315f3 b8b48991 8ef109e1 56193951 ec7e937b 1652c0bd 3bb1bf07 3573df88 3d2c34f1 ef451fd4 6b503f00
- G_x = c6 858e06b7 0404e9cd 9e3ecb66 2395b442 9c648139 053fb521 f828af60 6b4d3dba a14b5e77 efe75928 fe1dc127 a2ffa8de 3348b3c1 856a429b f97e7e31 c2e5bd66
- G_y = 118 39296a78 9a3bc004 5c8a5fb4 2c7d1bd9 98f54449 579b4468 17afbd17 273e662c 97ee7299 5ef42640 c550b901 3fad0761 353c7086 a272c240 88be9476 9fd16650

採用廣義梅森數

- 因為質數體橢圓曲線密碼系統的軟體實現常要計算

$$B = A \bmod p\text{，}0 \le A \le p^2$$

- 而p為大整數(例如192位元)，所以公式通常需計算兩個大整數的除法，而這個$A \div p$的除法運算速度很慢。然而NIST所推薦的5條橢圓曲線的質數體p都是某種特別形式，稱為廣義梅森數(generalized Mersenne number)，這種質數可加速公式的計算，簡化計算過程 詳見書本
- $p = 2^{192} - 2^{64} - 1$
- $p = 2^{224} - 2^{96} + 1$
- $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
- $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
- $p = 2^{521} - 1$

OpenSSL進行ECC

- 2005年7月所釋出的0.9.8版OpenSSL開放原始碼軟體新增橢圓曲線密碼功能，且內建多種不同橢圓曲線，可用來執行ECDH及ECDSA的功能
- 下列指令列出所有OpenSSL的內建ECC金鑰參數

  openssl ecparam –list_curves

OpenSSL測試ECDH效率

- 測試不同ECC參數的ECDH金鑰交換演算法效率指令: openssl speed ecdh

```
                                op      op/s
160 bit ecdh (secp160r1)   0.0047s    213.6
192 bit ecdh (nistp192)   0.0046s    218.0
224 bit ecdh (nistp224)   0.0067s    150.2
256 bit ecdh (nistp256)   0.0119s     84.3
384 bit ecdh (nistp384)   0.0269s     37.2
521 bit ecdh (nistp521)   0.0889s     11.2
163 bit ecdh (nistk163)   0.0052s    192.9
233 bit ecdh (nistk233)   0.0094s    106.5
283 bit ecdh (nistk283)   0.0188s     53.1
409 bit ecdh (nistk409)   0.0417s     24.0
571 bit ecdh (nistk571)   0.1018s      9.8
163 bit ecdh (nistb163)   0.0057s    176.7
233 bit ecdh (nistb233)   0.0107s     93.2
283 bit ecdh (nistb283)   0.0192s     52.1
409 bit ecdh (nistb409)   0.0441s     22.7
571 bit ecdh (nistb571)   0.1154s      8.7

C:\openssl\OUT32>
```

OpenSSL測試ECDSA效率

- 測試不同ECC參數的ECDSA數位簽章演算法效率指令: **openssl speed ecdsa**

```
                              sign    verify    sign/s verify/s
 160 bit ecdsa (secp160r1)   0.0011s   0.0056s   882.1   178.6
 192 bit ecdsa (nistp192)   0.0012s   0.0055s   858.1   180.6
 224 bit ecdsa (nistp224)   0.0016s   0.0076s   643.2   131.2
 256 bit ecdsa (nistp256)   0.0027s   0.0143s   374.8    69.9
 384 bit ecdsa (nistp384)   0.0058s   0.0309s   172.6    32.3
 521 bit ecdsa (nistp521)   0.0159s   0.0883s    62.8    11.3
 163 bit ecdsa (nistk163)   0.0035s   0.0097s   282.4   103.0
 233 bit ecdsa (nistk233)   0.0070s   0.0162s   142.0    61.7
 283 bit ecdsa (nistk283)   0.0109s   0.0318s    91.7    31.5
 409 bit ecdsa (nistk409)   0.0267s   0.0686s    37.4    14.6
 571 bit ecdsa (nistk571)   0.0636s   0.1702s    15.7     5.9
 163 bit ecdsa (nistb163)   0.0033s   0.0097s   307.3   103.1
 233 bit ecdsa (nistb233)   0.0068s   0.0178s   146.1    56.1
 283 bit ecdsa (nistb283)   0.0109s   0.0352s    91.6    28.4
 409 bit ecdsa (nistb409)   0.0268s   0.0782s    37.4    12.8
 571 bit ecdsa (nistb571)   0.0645s   0.1973s    15.5     5.1

C:\openssl\OUT32>_
```

參考資料

- Crypto++, http://www.cryptopp.com/
- D. Johnson and A. Menezes, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," Centre for Applied Cryptographic Research, University of Waterloo, August 1999. http://www.cacr.math.uwaterloo.ca/techreports/1999/corr99-34.pdf
- LiDIA, http://www.informatik.tu-darmstadt.de/TI/LiDIA/
- MIRACL, http://www.shamus.ie/
- NIST, *Digital Signature Standard (DSS)*, FIPS 186-3, June 2009, http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf
- NIST, *Recommendation for Key Management - Part 1: General*, Special Publication 800-57 Revision 4, January 2016, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf
- PARI/GP, http://pari.math.u-bordeaux.fr/