東海大學應用數學研究所

碩士論文

藉著橢圓曲線分解因數之研究

Factorization with Elliptic Curves

研 究 生：劉慕飛 撰

指導教授：沈淵源 博士

中華民國 105 年 5 月

誌謝

首先要感謝的是我的指導教授沈淵源老師不厭其煩耐心指導我每一個細節，不論是學問或是為人處世都給我很大的啟發。感謝百忙之中參與我口試的劉康滿教授陳淑珍教授。另外感謝淑珍老師讓我以另一個角度思考數學，感謝王道明老師為我的數論奠定基礎，感謝我的家人在我落寞失意之際給我支持，感謝我的母校華盛頓中學給我時間讓我繼續進修，最後感謝我的學長同學學弟沒有你們的幫忙就沒有現在的我。希望能將所學繼續在社會上、教育的領域幫助更多需要幫助的人，為自己為大眾盡一份心力。

劉慕飛　　謹識於

東海大學應用數學研究所

中華民國 105 年 5 月

摘要

這篇論文是研究藉由橢圓曲線來因數分解之間會遇到的問題，我們使用的軟體是 Mathematica10，將演算法寫成語言之後只需要輸入要分解的數字，並選取橢圓曲線即可。有些曲線對於分解數字所需要的計算時間較長，這時候我們必須選取另一條曲線，而新的這條曲線有機會以更快的速度分解同一個數字，或者以疊代的方式也能順利降低計算複雜度。該如何選曲線以及設計出更快的演算法是個值得探討的問題，在這一篇論文後仍有進步的空間。

目錄

第一章　前言

密碼學(Cryptology)爲確保資訊的保密書寫和傳遞的學問，在密碼系統中最有名的莫過於RSA加密演算法，我們根據RSA系統奠定在因數分解不易的條件下，提出如何將因數分解演算法之理論，以一般電腦運算幫助我們達到因數分解的目的。因數分解是單純基本、但計算不易的數論問題。現代的因數分解演算法皆有一個共通點，那就是分解基底的運用。一般來說，分解基底是以較少的質數形成基底，隨著各種演算法有各自的分解基底選擇方法。在下個章節我們會介紹密碼學的基本概念以及RSA加密演算系統與因數分解的關聯。第三個章節接著介紹現有的幾個因數分解演算法。第四章介紹橢圓曲線基本運算性質與模下的橢圓曲線。第五章以個人電腦實際運用橢圓曲線的程式分解大數，並探討其中遭遇的問題與發現。

第二章　密碼學的概念

這個章節我們會介紹密碼系統的產生，接著介紹 RSA 加密演算法，最後說明密碼系統與 RSA 演算法對於因數分解的相關性。

2.1 密碼學(Cryptology)

密碼學(Cryptology)是一種隱密的訊息傳遞之學問，目標是只讓特定的收信方解讀信息內容。未經加密的訊息內容我們稱之為明文(Plaintext、Cleartex)，而加密過後不易解讀的訊息我們稱為密文(Ciphertext)兩者皆以數個字元來表達，訊息可以以任何字元甚至符號的形式存在。明文與密文之間可以互相轉換，由明文轉換為密文的過程稱為加密(Enciphering or Encryption)，而由密文轉換為明文的過程我們稱為解密(Deciphering)，若將密碼系統的明文與密文看成兩組訊息的集合(通常兩個集合的元素個數會一樣，因為明文與密文多為一對一關係)，而加密看成函數，那麼解密就是加密的逆運算也就是反函數。這樣的系統我們就稱為密碼系統(Cryptosystem)。

2.2 RSA 公開加密演算法

RSA 加密演算法是一種加密容易破解難的演算法，也就是非對稱性密碼系統(Asymmetric Cryptosystem)是一種利用歐拉函數以及同餘性質產生一對公鑰與私鑰的一種加密方式。在公開金鑰加密和電子商業中 RSA 扮演重要的腳色。RSA 是 1977 年由 Ron Rivest、Adi Shamir 和 Leonard Adleman 一起提出的。RSA 取自其三人姓名之首，當時三人正就讀於麻省理工學院。RSA 簽署的安全性是奠定在兩個大位元質數的乘積不易因數分解，因此解密的關鍵在於如何破解兩個大位元質數的乘積。

2.2.1 RSA 簽署過程公鑰、私鑰的產生

假設 A 想要通過一個開放的平台接收 B 的一則私訊。下面是產生公鑰和私鑰的流程：

(1)任意挑選兩個較大質數 p 和 q（p、q 的數值不宜太過接近否則較容易遭到破解)， p 不等於 q，並計算 $n=pq$。

(2)根據 Euler’s Function，求得 $\varphi(n)=\varphi(p)\varphi(q)=(p-1)(q-1)$

選擇一個 e , $e<r, e\in Z$ ，使 $(\varphi(n),e)=1$找出 ， $e^{-1}=d(\mathrm{mod}\,\varphi(n))$

$ed\equiv 1(\mathrm{mod}\,\varphi(n))$ 。將 p 和 q 的記錄銷毀。

其中 (n,e) 為公鑰 ，(n,d) 則為私鑰。A 將他的公鑰 (n,e) 傳給 B，並將他的私鑰 (n,d) 藏起來。

加密訊息

假設 B 傳給 A 一個訊息 X，他知道阿呆產生的 (n,e)。他使用原先與 A 約定好的格式將 X 轉換為一個小於 n，且與 n 互質的整數 Y，假設訊息量過大可以分段將訊息轉換成數字碼。用下面這個公式他可以將 X 加密為 Y：

$$Y \equiv X^e \pmod{n}$$

A 算出 Y 後就可以將它傳遞給 B。

解密訊息

B 收到 A 加密後的訊息 $Y \equiv X^e \pmod{n}$ 後就可以利用他的私鑰 (n,d) 來解碼。

只要算出 e 在 $\bmod \varphi(n)$ 底下的乘法反元素，即可將密文還原成明文

$$X \equiv Y^d \pmod{n}$$

2.2.2 因數分解與 RSA 的關聯

對於 RSA 演算法來說公鑰 (n,e) 是公開的，私鑰 (n,d) 是解密者自己持有的密鑰，若想得到私鑰 (n,d) 我們必須對 n 進行因式分解進而得到 p、q 再算出 d 的值即可，但是話雖容易，要分解 RSA 簽署的 n 動

輒百位大數，有些甚至到達千位，就算我們知道一個合數一定能被分解但是需要時間成本，以密碼學的角度來看這個 RSA 系統還是一個安全的系統，是因為大部分訊息有時效性，要是能在一定的時間內保密，目的就算達成，因此若想解密我們需要更有效率的演算法才可能在限制的時間內將此合數分解。

第三章　因數分解的演算法

數論是個古老卻又新生的學科，數學的源起就是離散數學，而有些數論問題被棄置多年直到近代電腦快速發展，很多繁瑣的計算可以交給程式執行，又重新賦予數論新的生命。以下介紹幾個因數方解法使用到的定理:

3.0.1 費馬小定理

若 P 為質數且 $(a,P)=1$，則在模 P 之下，a 的 $P-1$ 次方和1同餘。

$a^{P-1} \equiv 1 (\mathrm{mod}\, P)$

3.0.1 歐拉函數

歐拉函數表示所有小於正整數 n 且與 n 互質的正整數個數。

$\varphi(n) = \prod_{k=1}^{\infty}(1-q^k)$

根據上述定義可以衍伸以下性質:

(1) 若 P 為質數則 $\varphi(n) = p-1$

(2) 若 $n = pq, p \neq q$ 則 $\varphi(p)\varphi(q) = (p-1)(q-1)$

(3) 若 P 為質數，且 k 為自然數則 $\varphi(p^k) = p^k - p^{k-1} = p^k(1-\frac{1}{p})$

3.0.2 歐拉定理

在模 n 之下 a 的歐拉數次方和1同餘。

$a^{\phi(n)} \equiv 1 (\mathrm{mod}\, n)$

3.1 試除法

最簡單最直接的大數分解方法就是試除法，也就是不斷嘗試取介於2與$\sqrt{n}$之間的質數來當除數，當數值很小的時候，此方法可以容易地分解出結果來，然而當數字很大的時候除非有兩相近質因數，否則從 2 開始，一直算到$\sqrt{n}$需要 $\pi(\sqrt{n})$次試除，這裡的$\pi(x)$是小於x的質數的個數。

3.2 費馬演算法

若$n=ab$ 且$a \geq b > 0$, 則

$$n=ab=[\frac{1}{2}(a+b)]^2-[\frac{1}{2}(a-b)]^2=t^2-s^2$$

$t=[\sqrt{n}]+1$ 開始找，直到找到t^2-n為完全平方數即可找到s ，費馬演算法適用於兩個因數都很靠近$\sqrt{n}$ ，雖然如此卻奠定了因數分解的基本概念，狄克森演算法(Dixon’s Algorithm)也是運用類似的觀念尋找平方關係。

3.3 克雷奇克演算法

若$n \mid a^2-b^2$ 則$a^2 \equiv b^2 (\mathrm{mod}\, n) \Rightarrow n \mid a^2-b^2 = n \mid (a+b)(a-b)$

在此條件下會有兩種可能的情況

1. $n \mid a+b$ 且$n \mid a-b$ 則$\gcd(n, a \pm b) = n$ 無法用來分解n

2. n 不為 $a+b$ 之因數或 n 不為 $a-b$ 之因數，

在此條件下 $\gcd(n,a+b)<n$ 或 $\gcd(n,a-b)<n$，換句話說 $\gcd(n,a+b)$ 或 $\gcd(n,a-b)$ 為 n 的因數這樣的 a,b 就可以用來分解 n，找 a,b 的方法為:

$Q(x)=x^2-n$ 隨意的找 x 使得 $Q(x)$ 的連乘積為平方數，若

$Q(x_1)\cdots Q(x_k)=b^2, x_1\cdots x_k=a$ 則

$$a^2=x_1^2\cdots x_k^2\equiv(x_1^2-n)\cdots(x_k^2-n)=Q(x_1)\cdots Q(x_k)=b^2 \pmod n$$

我們發現如果一個較大的平方數不易找到時，可以利用許多不同小質數的次方和為偶數，使其乘積為平方數；這些質數便稱為分解基底(Factor base)。

3.4 連分數分解法

對於每一個 $x\in R$，我們都可將 x 寫成連分數的型態:

令 $a_0=[x], x_0=x-a_0$，令 $a_1=[\frac{1}{x_0}], x_1=\frac{1}{x_0}-a_1$ 當 $i>1$ 令 $a_i=[\frac{1}{x_i-1}]$，直到得到 $[\frac{1}{x_1-1}]\in Z$。當 $x\in Q$ 時

$$x=a_0+\cfrac{1}{a_1+\cfrac{1}{a_2+\cdots+\cfrac{1}{a_i+x_i}}}$$

或是整理成 $x=a_0+\frac{1}{a_1+}\frac{1}{a_2+}\cdots\frac{1}{a_i+x_i}$

演算步驟:

設 n 為一合數，令 $b_{-1}=1, b_0=a_0=[\sqrt{n}], x_0=\sqrt{n}-a_0$ 算出 $b_0^2 \pmod n$ ，

接著令 $i=1,2,\cdots$ ，依序算出

(1) $a_i=[\frac{1}{x_i-1}]$ 、 $x_i=\frac{1}{x_{i-1}}-a_i$

(2) $b_i=a_i b_{i-1}+b_{i-2}$ 令 $c_i \equiv b_i \pmod n$

(3) 算出 $b_i^2 \pmod n$

直到找到 $\{c_i \mid k=1,2,\cdots\}$ 其中的元素相乘為平方數。

3.5 二次篩法

定義下列多項式:

$Q(x)=(x+\lfloor\sqrt{n}\rfloor)^2-n$ ，對於任意的 x

$|Q(x)| \le (|x|+\sqrt{n})^2-n=|x|^2+2|x|\sqrt{n}+n-n=|x|(|x|+2\sqrt{n})$

當 x 的值很小時 $|Q(x) \approx 2\sqrt{n}|$ 且

$Q(x)=(x+\lfloor\sqrt{n}\rfloor)^2-n \equiv Q(x)=(x+\lfloor\sqrt{n}\rfloor)^2 := F(x) \pmod n$

也就是說我們可以利用 $Q(x) \equiv F(x) \pmod n$ 來建立許多形如 $x \equiv y^2 \pmod n$ 的同餘式，$Q(x)$ 的值不會過大且 $F(x)$ 為平方數。

然後再利用預選的分解基底(factor base)對 $Q(x)$ 進行分解。分解基底的集合是由許多較小質數所組成，這些質數都不會超過 B ，假設 $Q(x)$ 為 B-*smooth*（表此整數的所有因數都小於等於 B 則稱 B-*smooth*），

則 n 為 $\bmod p_i$ 下的二次剩餘，分解基底的集合為:

$\{p_i \mid p_i \in prime, p_i \le B, (\frac{n}{p_i}) = 1\}$ ，若

$$k \in Z \Rightarrow Q(x+kp_i) = (x+kp_i+\lfloor \sqrt{n} \rfloor)^2 - n = ((x+\lfloor \sqrt{n} \rfloor)+kp_i)^2 - n \equiv (x+\lfloor \sqrt{n} \rfloor)^2 - n (\bmod p_i)$$

由上述式子可知若 $p_i \mid Q(x) \Rightarrow p_i \mid Q(x+kp_i)$ 可檢測 $Q(x)$ 是否為 B-*smooth* 。

$Q(x)=(-1)^{e0}\prod_{i=1}^{f} p_i^{e_i}, p_i$ 為 factor base 中的元素，其中 $e_i \in Z, e_i \ge 0$ ，我們可將 e_i 用向量表示 $\vec{v}(x) = (e_1, e_2, \cdots e_f)(\bmod 2)$ 。

找到足夠多的向量後，建構一個 $Z/2Z$ 的矩陣，以高斯消去法求得一組線性相依:

$\vec{v}(x_1) + \cdots \vec{v}(x_j) = 0$ 使得 $Q(x_1)\cdots Q(x_k) = b^2$ ，算出 $b(\bmod n)$ 和 $a = \sqrt{F(x_1)\cdots F(x_k)}(\bmod n)$ ，即可得到

$$a^2 \equiv b^2 (\bmod n) \Rightarrow n \mid a^2 - b^2 = n \mid (a+b)(a-b)$$

就可以用來因式分解。

3.6 數體篩法

數體篩法是利用代數數體與有理整數來和平方關係取得關聯，用以分解 $r^e \pm s$ 這類型的有理整數， s,r 皆為較小的整數，冪次 e 較兩者為大。

若目標n為想分解的有理整數且nk可寫成$r^e \pm s$的形式，比如第七個費馬數，$2F_7 = 2(2^{128}+1) = 2^{129}+2 = m^3+2, m = 2^{43}$ 利用一個領導係數為 1 的三次多項式 $f(x) = x^3 + 2 \equiv 0 (\mathrm{mod}\, n)$ ，若 $f(\alpha) = 0$ 且 α 為複數，根據代數整數環 $Z[\alpha]$ 的性質，所有 $Z[\alpha]$ 中的元素都可表示為 α 的多項式，且各項係數均為有理整數。

由 $f(\alpha) = 0, f(m) \equiv 0 (\mathrm{mod}\, n)$ 可找到一個由整數環 $Z[\alpha]$ 對應到 Z_n 的環同態 ϕ ，且 $\phi(\alpha) = m (\mathrm{mod}\, n)$

$\alpha = Z[\alpha] \to Z_n$

$\alpha \mapsto m$

找一個有限集合 $S = \{ < a_i, b_i > | \gcd(a_i, b_i) = 1, i = 1, 2, \cdots k \}$ 並使數對滿足：

(1) $\prod_{i=1}^{k} \phi(a_i - \alpha b_i)$ 在整數 $Z[\alpha]$ 中為平方數 $= \gamma^2$

(2) $\prod_{i=1}^{k} \phi(a_i - m b_i)$ 在有理整數 Z 中為平方數 $= \nu^2$

令 $\phi(\gamma) \equiv u (\mathrm{mod}\, n)$

帶入(1)可得

$$u^2 \equiv (\phi(\gamma))^2 = \phi(\gamma^2) = \phi(\prod_{i=1}^{k} (a_i - \alpha b_i)) = \prod_{i=1}^{k} \phi(a_i - \alpha b_i)) \equiv \prod_{i=1}^{k} \phi(a_i - m b_i) = \nu^2 (\mathrm{mod}\, n)$$

即成功找到平方關係。

另外分解演算法較著名的還有波拉德演算法(Pollard's rho algorithm)、波拉德 P-1 演算法(Pollard's P-1 algorithm)、威廉 P+1 演算法(Williams p+1)等構想理念大同小異就不多做介紹。

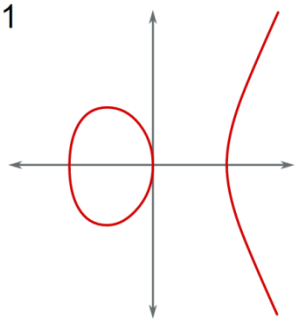
第四章 橢圓曲線

由 Hendrik Lenstra 所提出，利用橢圓曲線來分解因數，是目前被公認為最佳的因數分解演算法，在 $n=pq$ 的條件下以平方關係因式分解的演算法會比橢圓曲線法更容易分解 p,q 靠近 $\sqrt{n}$ ，因此橢圓曲線更適合找到較小的因數，也就是說橢圓曲線分解的難度是基於 p,q 中較者的大小若令 q 為兩者中較小的則 q 值越小難度越低，q 值越大分解難度越高。

4.1 橢圓曲線方程式

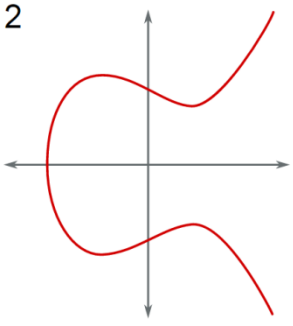
橢圓曲線方程式形如 $E: y^2+a_1xy+a_3y=x^3+a_2x^2+a_4x+a_5$ 我們可經由簡單的變數變換將曲線整理成 $E: y^2=x^3+ax+b$ 、 $E: y^2=P(x)$ ，a、b 佈於任何適用之集合如有理數、實數、複數、模 n 下或者任何有限數體，P 為沒有重根的三次多項式。下列圖 1 為三實根的橢圓曲線，圖 2 則為一實根兩虛的根的橢圓曲線，圖 3 則為不同的 a,b 值對應的橢圓曲線

1

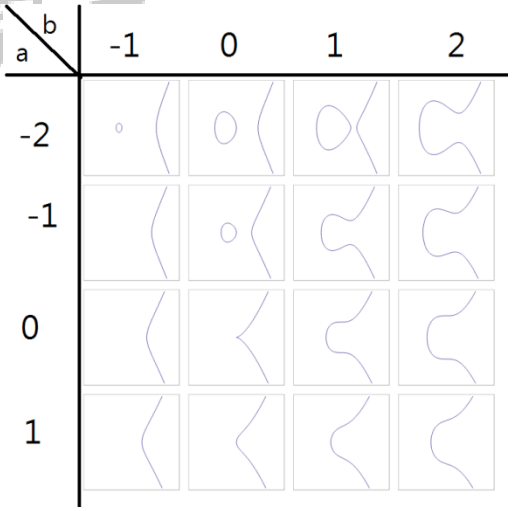


$y^2=x^3-x$

2



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



4.2 橢圓曲線上的加法運算

定義橢圓曲線上的加法運算，一般討論形如以下的橢圓曲線:

$E: y^2 = x^3 + ax + b$ 在 E 上給予兩點 P_1 及 P_2 可得到第三點 P_3，作法是畫一條過 P_1 及 P_2 的直線 L，直線 L 交 E 於 Q 點則 P_3 即為 Q 對稱於 X 軸的對稱點可寫成 $P_1+P_2=P_3$ 或是 $P_1+P_2+Q=\infty$；若 $P_1=P_2$ 則以過此點切線為 L，也就是 $P_1+P_2=2P_1$ 且 $Q+2P_1=\infty$

4.3 橢圓曲線上的無窮遠點∞

∞ 為投影平面上的無窮遠點，可以想像座標 (a,b) 當 $b \to \pm\infty$ 都在這點上，所以經過 P 點及 ∞ 的直線是一條鉛直線，此線與 E 交於 $P(x,y),\infty,(x,-y)$ 三點，而點 $(x,-y)$ 與 X 軸之對稱點就是 $P(x,y)$ 點，因此得到 $P+\infty=P$，換句話說∞為橢圓曲線上的加法單位元素。 下面的圖可以幫助我們更了解加法運算。

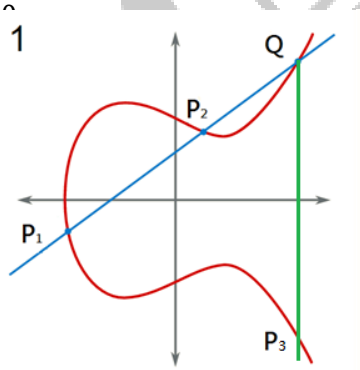


圖 1　$P_1+P_2=P_3$

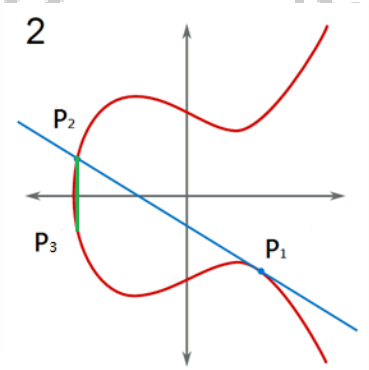
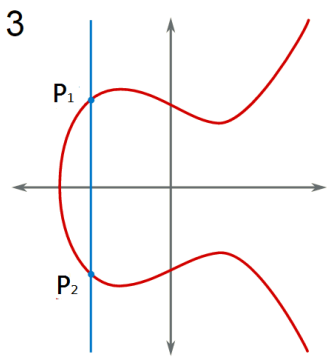


圖 2　$2P_1=P_3$

圖 3　$P_1+P_2+\infty=\infty$



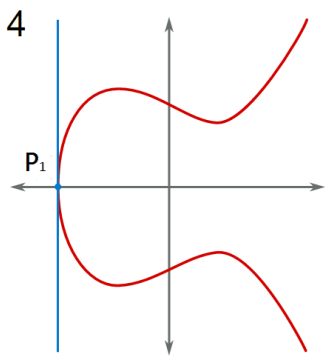


圖 4　$P_1+P_1+\infty=\infty$

4.4 橢圓曲線加法運算式

若 $E: y^2 = x^3 + ax + b$ 且 $P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_3 = (x_3, y_3)$ 公式將 $P_1+P_2=P_3$ 表示出來 $x_3 = m^2 - x_1 - x_2$ ， $y_3 = -[m(x_3 - x_1) + y_1]$ ， m 為 P_1,P_2 連線的斜率:

$$m = \begin{cases} (y_2 - y_1)/(x_2 - x_1) & P_1 \neq P_2 \\ (3x_1^2 + a)/(2y_1) & P_1 = P_2 \end{cases}$$

模 n 下的橢圓曲線 $E: y^2 \equiv x^3 + ax + b \pmod n$ 就是所有滿足此方程式的數對 (x, y) 以及一個無限遠點 ∞，所以在模 n 下的橢圓曲線是一堆分散的點，點與點之間的和也可以用上述公式來求得，特別的是當我們遇到的斜率形如有理數 $\frac{s}{t}$ 則必須尋找 t 的乘法反元素也就是 $\frac{s}{t} = st^{-1}$ ，

且 t 的乘法反元素 t 和 n 必須互質也就是 $\gcd(t, n) = 1$ ，才能求得 t^{-1} 。

例題 1: 求 $E: y^2 \equiv x^3 + 2x + 3 \pmod 5$ 上點 $(1,4)+(3,1) = (x_3, y_3)$

E 上的點就是所有滿足此方程式模 5 底下的數對 $(x, y) \pmod 5$ 以及 ∞

可能的 x 之值為 0、1、2、3、4、5，將每個值代入方程式中求出對應的 y 值我們可以得到以下幾點 $(1,1)(1,4)(2,0)(3,1)(3,4)(4,0)$ ∞ ，

$m = \frac{1-4}{3-1} = \frac{-3}{2}$ 時我們需要尋找 $\frac{1}{2} \equiv x \pmod 5$ 即 $1 \equiv 2x \pmod 5$ 因為 $\gcd(2,5) = 1$ 我們可以找模 5 之下 2 的乘法反元素，得到 $x = 3$ 。

$$m \equiv \frac{1-4}{3-1} \equiv \frac{-3}{2} \equiv -3 \times 3 \equiv 1 \pmod 5$$

因此 $x_3 \equiv m^2 - x_1 - x_2 \equiv 1^2 - 1 - 3 \equiv 2 \pmod 5$

且 $y_3 \equiv m(x_1 - x_3) - y_1 \equiv 1(1-2) - 4 \equiv 0 (\bmod 5)$ 故得到 $(1,4)+(3,1) = (2,0)$

例題 2: 求 $E : y^2 \equiv x^3 + 4x + 4 (\bmod 2773)$ 上點 $(1,3)+(1,3) = (x_3, y_3)$

首先計算此曲線在點 $(1,3)$ 上的切線斜率:

$$2ydy = (3x^2 + 4)dx \Rightarrow \frac{dy}{dx} = \frac{7}{6}$$

接著計算 6 的乘法反元素，因為 $\gcd(6, 2773) = 1$，

利用輾轉相除法我們可以將 1 寫成 6 與 2773 的一個線性組合:

$1=(-462)\times 6+1\times 2773$

因此 $\frac{1}{6} \equiv -462 \equiv 2311 (\bmod 2773)$ 最後得到:

$$m \equiv \frac{7}{6} \equiv 7 \times 2311 \equiv 2312 (\bmod 2773)$$

由公式解得:

$$x_3 \equiv m^2 - x_1 - x_2 \equiv 2312^2 - 1 - 1 \equiv 1771 (\bmod 2773)$$

$$y_3 \equiv m(x_1 - x_3) - y_1 \equiv 2312(1 - 1771) - 3 \equiv 705 (\bmod 2773)$$

故得到 $(1,3)+(1,3) = (1771, 705)$

第五章橢圓曲線分解因數法

而橢圓曲線分解因數的關鍵就在於要找到前一章提到t的乘法反元素需要$\gcd(t,n)=1$，所以當我們不斷的對同一點做加法的同時t的值會隨之變動直到兩點的x座標相同，讓斜率分母為0，也就是找不到模n下t的乘法反元素，而此時正是$\gcd(t,n)\neq 1$也就是說n與t有除了1以外的其他因數，那麼n也就隨之分解。

例題 3: 求 $E: y^2 \equiv x^3+4x+4 \pmod{2773}$上 $(1,3)+(1,3)+(1,3)=(x_3,y_3)$

我們以例題 2 的結果繼續計算 $(1770,705)+(1,3)=(x_3,y_3)$ ， $m=\frac{702}{1770}$

接著計算1771在模2773下的乘法反元素，卻因$\gcd(1771,2773)\neq 1$無法得知結果，將1771與2773以輾轉相除法運算可以得到$\gcd(1771,2773)=59$進而找到$2773=59\times 47$ 。

5.1 定義程式碼

以下我們用 Mathematica10 將加法運算寫成三種指令:

(1) ecadd[p1_,p2_,a_,b_,n_]

程式見附錄 1，表示在模n下將橢圓曲線$E: y^2=x^3+ax+b$上的兩個點P_1及P_2做加法運算，例如輸入 ecadd[{1,5},{3,3},2,3,19]

我們將橢圓曲線$E: y^2 \equiv x^3+2x+3 \pmod{19}$上做點(1,5),(3,3)的加法得到的結果就是(6,10)，

而輸入 ecadd[{1,5},{1,3},2,3,19]會得到的是{infinity, infinity }這是因為(1,5)和(1,3)的x座標相同而此時連線斜率分母為0為一條鉛直線，所以得到(1,5)+(1,3)=∞

(2)ecmus[p1_,m_,a_,b_,n_]

程式見附錄 2，表示做P_1的m倍並列出每一次加法的結果:

輸入 ecmus[{1,5},10 ,2,3,19]我們得到

(1,5)(3,13)(12,8)(10,15)(9,3)(15,8)(14,18)(5,10)(11,11)(18,0)

就是對(1,5)做9次加法並列出所有情形。

(3)ecmlt[p1_,m_,a_,b_,n_]

程式見附錄 3，是列出mP_1的結果:

例如輸入 ecmlt[{1,5},10 ,2,3,19]我們得到

(18,0)

5.2 實際分解因數

例題 4:因式分解東海大學電話 23590121

我們可以藉由 ecmus[p1_, m_, a_, b_, n_]ecmlt[p1_, m_, a_, b_, n_]

找到n的因數，有了目標n之後找選擇一點當作 P1 代入

$E: y^2 \equiv x^3 + ax + b \pmod{n}$再決定$a$的值最後將$b$計算出來就可完成準備工作。在此我們選擇一點(3, 2)，並令a為 0 可算出b的值為-2，

$2^2 \equiv 3^3 + 0 \times 3 - 2 \pmod{23590121}$確定點(3, 2)在這條曲線上後我們可以

輸入 ecmlt[{3,2},435 ,0,-2,23590121]

得到(2273644,13508616)

輸入 ecmlt[{3,2},436 ,0,-2,23590121]

得到的是{factor,1753 }，23590121 成功分解。

這是因為當我們做 435P 時得到一點

(2273644,13508616)這時尚未發生錯誤，還未分解因數。

而當我們做 436P 時因為斜率分母為0且找不到t^{-1}，這使得

$\gcd(t,23590121)\neq 1$進而分解出因數 1753。我們將 23590121 的兩個因數 1753、13457 獨立出來探討。就可以觀察出在因數被分解出的當下發生了什麼事情。

輸入 ecmus[{3,2},436 ,0,-2,1753] ，執行結果見附錄 4

輸入 ecmlt[{3,2},436 ,0,-2,13457]

輸出(1993,1418)，此點與(3,2)之連線不為垂直線。

也就是說在模 1753 下，最後三點間的運算為(3,1751)+(3,2)=∞，此時 x 座標相同斜率分母為 0 亦即找不到t的乘法反元素，而在模 13475 下加法運算還是得以持續進行。這時候因數 1753 可以被找到。

在正常情況下我們有很高的機率單單以幾條曲線就能將$n=pq$分解，

而其中又以 p,q 中較小的質因數會先被找到，這是因為在模的運算中較小的模之下會使得 $P_i(x_i,y_i)$ 中的 x 座標重複的機率較高，也會使得斜率分母為0且找不到 t 的乘法反元素，進而找到因數。

5.3 以疊代法降低計算複雜度

有時候分解一個大數會因計算複雜度過高，導致計算時間太長無法順利找到因數，很可惜的是以 Mathematica10 來說我們無法進行水平運算，也就是說一次只能試一條曲線，除非同時開啟多台電腦，不過這種方法既耗時間又耗資源，我們需要更快更有效率的演算法來改善這種情況，疊代法就是很好的選擇。

疊代法的指令:

```
p[1]={x,y};p[i_]:=ecmlt[p[i-1],i,a,b,n]
Table[p[i],{i,k}]//ColumnForm
```

此處的做法是以階乘來進行點的加法，P_i 被定義為前一個點 P_{i-1} 做 i 次加法，也就是計算點 $1!P$ 、$2!P$ 、$3!P$ 、$4!P$ ，若我們計算到 $k!P$ 實際上我們只做 $\frac{(1+k)k}{2}$ 次加法，這種做法可以省去逐點做加法的麻煩。

例題 4: 以疊代法分解 23590121

```
p[1]={3,2};p[i_]:=ecmlt[p[i-1],i,0,-2,23590121]
Table[p[i],{i,110}]//ColumnForm
```

，執行結果見附錄 5

如附錄 5 資料我們得到在計算 $110!P$ 時得到因數 1753，而由原本逐點計算的方式我們會在計算 $436P$ 時得到因數 1753 也就是說以階乘為

加法方式有機會錯過原先可以分解的點，但最終還是會碰到其他使得n可以分解的其他點

例題 5:分別以逐點的方式以及疊代法分解 3941046533

```
ecmlt[{1,5},12! ,2,3,3941046533]
{{"factor=", 52673}}

p[1]={3,2};p[i_]:=ecmlt[p[i-1],i,0,-2,3941046533]
Table[p[i],{i,1000}]//ColumnForm
{{"factor=", 74821}}
```

例題6:分解10924475509

```
ecmlt[{1,5},12! ,2,3, 10924475509]
{{"factor=", 111271}}
ecmlt[{1,1},12! ,3,-3, 10924475509]
{{"factor=", 98179}}
```

例題8:分解451375011285641

```
ecmlt[{1,2},12! ,17,-14, 451375011285641]
{{"factor=", 221471}}
```

例題9:分解7106530910783693

```
ecmlt[{1,-2},12! ,17,-14, 7106530910783693 ]
{{"factor=", 98899}}
```

例題10:分解10221054713191529

```
ecmlt[{1,-2},12! ,7,-4, 10221054713191529]
{{"factor=", 105359}}
```

例題11:分解612105697527308183

```
ecmlt[{1,-2},12! ,7,-4, 612105697527308183]
{{"factor=", 221719}}
```

例題11:分解612105697527308183

```
p[1]={2,2};p[i_]:=ecmlt[p[i-1],i,0,-4,37640901318
1357107916529]
Table[p[i],{i,10!}]//ColumnForm
```

```
{{"factor=", 6461338097}}
```

例題 12 分解 19776383191770192017781219836831932879 49

```
p[1]={3,2};p[i_]:=ecmlt[p[i-1],i,0,-2,
19776383191770192017781219836831932879 49]
Table[p[i],{i,1000}]//ColumnForm
```

```
{{"factor=", 439734563409764534 57}}
```

5.4 結論

總結以上各點因式分解 n 的步驟為:

(1) 選取任意橢圓曲線滿足 $E: y^2 = x^3 + ax + b$ ，其中先選定一點 $P_1 = (x_1, y_1)$ 再選擇 a 的值最後將 b 值計算出來。

(2) 逐點對 P_1 執行數次加法直到將 n 分解。

(3) 若步驟(2)執行結果過長可以開啟多台電腦同時執行步驟(1)，只要選取的曲線夠多一定可以找到一條順利將 n 分解，或是進行下個步驟。

(4) 以疊代法跳點尋找 $1!P$ 、 $2!P$ 、 $3!P$ 、 $4!P$ ...藉此增加碰到可以分解點的機會。

(5) 另一種疊代法計算出最後一點 $k!P$ 再將此點重新代入 P_1 。

有時候以逐點的方式只藉由單單幾條曲線是沒辦法順利將其分解的，上述的方法可以讓我們更有效率的分解 n。模 n 下的橢圓曲線是由許多格子點構成的集合，其中有些點能使目標 n 分解，有些不能。想像一下若要蒙著眼睛在草地上尋寶，我們可以在不同的草原上一步一步走(選取不同曲線逐點尋找)，也可以改變不同的步伐在草原上前進(疊代法)。只要能更有效率的增加與草地接觸面積就能以更快的速度找到寶物。

附錄

附錄1程式

將P1P2相加

```
ecadd[p1_,p2_,a_,b_,n_]:=Module[{z,m,x3,y3,p3},z=0;z1=1;If[p1=={"infinity","
infinity"},p3=p2;z=1," "];
If[z==1," ",If[p2=={"infinity","infinity"},p3=p1;z=1," "]];
If[z==1,"
",If[p1[[1]]==p2[[1]]&&p1[[2]]==p2[[2]]==0,p3={"infinity","infinity"};z=1,"
"]];
If[z==1,"
",If[p1[[1]]==p2[[1]]&&p1[[2]]!=p2[[2]],p3={"infinity","infinity"};z=1," "]];
If[z==1,"
",If[p1==p2&&GCD[p1[[2]],n]!=1&&GCD[p1[[2]],n]!=n,z=1;z1=GCD[p1[[2]],n],"
"]];
If[z==1," ",If[p1==p2,m=Mod[(3*p1[[1]]^2+a)*PowerMod[2*p1[[2]],-1,n],n];
z=1;x3=m^2-p1[[1]]-p2[[1]];
y3=m*(p1[[1]]-x3)-p1[[2]];p3=Mod[{x3,y3},n]," "]];
If[z==1," ",If[GCD[p2[[1]]-p1[[1]],n]!=1,z=1;
z1=GCD[p2[[1]]-p1[[1]],n]," "]];
If[z==1," ",m=Mod[(p2[[2]]-p1[[2]])*PowerMod[p2[[1]]-p1[[1]],-1,n],n];
x3=m^2-p1[[1]]-p2[[1]];y3=m*(p1[[1]]-x3)-p1[[2]];
p3=Mod[{x3,y3},n]];If[z1==1,p3,{"factor=",z1}]];
```

附錄2程式

將m個P1做加法運算並顯示每個結果

```
ecmus[p1_,m_,a_,b_,n_]:=Module[{z},z={p1};

For[i=1,i<m&&z[[Length[z]]][[1]]!="factor=",i++,z=Append[z,ecadd[p1,z[[Lengt
h[z]]],a,b,n]]];z]
```

附錄3程式

將m個P1做加法運算後顯示最後一個結果

```
ecmlt[p1_,m_,a_,b_,n_]:=Last[ecmus[p1,m,a,b,n]]
```

附錄4執行結果

{{3, 2}, {1683, 929}, {1750, 258}, {457, 670}, {1667, 1658}, {1694, 1661}, {746, 1190}, {1399, 918}, {868, 872}, {1502, 294}, {448, 620}, {703, 1744}, {498, 1596}, {43, 1392}, {1052, 1673}, {1341, 131}, {1257, 844}, {877, 1536}, {1675, 1484}, {227, 915}, {551, 300}, {1496, 101}, {1208, 86}, {1404, 72}, {1093, 155}, {906, 1420}, {625, 1279}, {531, 915}, {1696, 540}, {1231, 1426}, {1095, 85}, {1494, 1132}, {151, 1747}, {1179, 867}, {51, 392}, {642, 1599}, {1137, 1139}, {908, 1312}, {209, 1201}, {34, 1511}, {1570, 287}, {855, 738}, {1144, 1218}, {1739, 159}, {254, 1388}, {1040, 1186}, {98, 586}, {995, 838}, {1507, 1049}, {1020, 1592}, {1298, 1438}, {804, 1698}, {276, 175}, {1115, 706}, {562, 1725}, {790, 1511}, {1467, 1207}, {133, 813}, {509, 644}, {1583, 266}, {1588, 532}, {762, 76}, {349, 1029}, {1102, 399}, {1425, 1601}, {1460, 427}, {1113, 277}, {433, 918}, {1121, 1473}, {497, 1631}, {107, 1039}, {775, 863}, {104, 805}, {1752, 429}, {766, 372}, {1674, 835}, {1617, 1472}, {626, 1151}, {1516, 964}, {136, 1393}, {608, 1302}, {564, 1183}, {230, 1295}, {1309, 266}, {82, 1480}, {1122, 942}, {412, 1268}, {1146, 1599}, {596, 1243}, {941, 1204}, {1348, 1003}, {338, 68}, {1544, 45}, {1710, 1229}, {638, 130}, {929, 242}, {529, 210}, {1217, 951}, {567, 617}, {831, 1333}, {283,1140}, {398, 83}, {200, 752}, {1073, 1744}, {1109, 1071}, {999, 801}, {1202, 1157}, {722, 399}, {1039, 311}, {930, 1263}, {337, 232}, {1407, 1393}, {1462, 971}, {1269, 128}, {1003, 1369}, {1730, 9}, {505, 268}, {965, 1353}, {983, 1315}, {692, 522}, {1272, 1554}, {543, 643}, {617, 1600}, {1718, 154}, {48, 178}, {166, 1542}, {1692, 807}, {161, 68}, {1267, 1223}, {966, 542}, {255, 725}, {66, 1132}, {137, 544}, {1101, 842}, {487, 1716}, {395, 131}, {442, 815}, {101, 699}, {521, 1072}, {438, 1333}, {1337, 1409}, {1409, 1456}, {936, 649}, {614, 1487}, {286, 125}, {718, 827}, {245, 393}, {1481, 522}, {57, 1630}, {998, 321}, {1165, 191}, {17, 1622}, {1385, 856}, {1488, 1531}, {991, 1370}, {958, 159}, {796, 165}, {462, 936}, {1428, 1097}, {120, 1698}, {1636, 1288}, {18, 597}, {189, 1385}, {193, 621}, {818, 1264}, {39, 865}, {369, 1450}, {1333, 522}, {1746, 885}, {561, 536}, {350, 533}, {1682, 1354}, {704, 1103}, {207, 1098}, {631, 680}, {810, 863}, {1565, 482}, {188, 931}, {942, 229}, {513, 670}, {378, 538}, {1244, 641}, {815,176}, {783, 1083}, {1025, 1310}, {1002, 1210}, {1265, 1248}, {168, 863}, {1480, 992}, {345, 615}, {637, 1527}, {210, 360}, {525, 772}, {31, 447}, {431, 1460}, {484, 1333}, {335, 1266}, {1216, 1019}, {502, 967}, {809, 1594}, {575, 395}, {1650, 837}, {1679, 731}, {1343, 72}, {403, 1076}, {1053, 1123}, {1569, 600}, {759, 72}, {471, 539}, {281, 1005}, {1433, 1321}, {829, 55}, {45, 233}, {1297, 1646}, {632, 1277}, {1254, 68}, {1092, 218}, {139, 0}, {1092, 1535}, {1254, 1685}, {632,476}, {1297, 107}, {45, 1520}, {829, 1698}, {1433, 432}, {281, 748}, {471, 1214}, {759, 1681}, {1569, 1153}, {1053, 630}, {403, 677}, {1343, 1681}, {1679, 1022}, {1650, 916}, {575, 1358}, {809, 159}, {502, 786}, {1216, 734}, {335, 487}, {484, 420}, {431, 293}, {31, 1306}, {525, 981}, {210, 1393}, {637, 226}, {345, 1138}, {1480, 761}, {168, 890}, {1265, 505}, {1002, 543}, {1025, 443}, {783, 670}, {815, 1577}, {1244, 1112}, {378, 1215}, {513, 1083}, {942, 1524}, {188, 822}, {1565, 1271}, {810, 890}, {631, 1073}, {207, 655}, {704, 650}, {1682, 399}, {350, 1220}, {561, 1217}, {1746, 868}, {1333, 1231}, {369, 303}, {39, 888}, {818, 489}, {193, 1132}, {189, 368}, {18, 1156}, {1636, 465}, {120,55}, {1428, 656}, {462, 817}, {796, 1588}, {958, 1594}, {991, 383}, {1488, 222}, {1385, 897}, {17, 131}, {1165, 1562}, {998, 1432}, {57, 123}, {1481, 1231}, {245, 1360}, {718, 926}, {286, 1628}, {614, 266}, {936, 1104}, {1409, 297}, {1337, 344}, {438, 420}, {521, 681}, {101, 1054}, {442, 938}, {395, 1622}, {487, 37}, {1101, 911}, {137, 1209}, {66, 621}, {255, 1028}, {966, 1211}, {1267, 530}, {161, 1685}, {1692, 946}, {166, 211}, {48, 1575}, {1718, 1599}, {617, 153}, {543, 1110}, {1272, 199}, {692, 1231}, {983, 438}, {965, 400}, {505, 1485}, {1730, 1744}, {1003, 384}, {1269, 1625}, {1462, 782}, {1407, 360}, {337, 1521}, {930, 490}, {1039, 1442}, {722, 1354}, {1202, 596}, {999, 952},

{1109, 682}, {1073, 9}, {200, 1001}, {398, 1670}, {283, 613}, {831, 420}, {567, 1136}, {1217, 802}, {529, 1543}, {929, 1511}, {638, 1623}, {1710, 524}, {1544, 1708}, {338, 1685}, {1348, 750}, {941, 549}, {596, 510}, {1146, 154}, {412, 485}, {1122, 811}, {82, 273}, {1309, 1487}, {230, 458}, {564, 570}, {608, 451}, {136, 360}, {1516, 789}, {626, 602}, {1617, 281}, {1674, 918}, {766, 1381}, {1752, 1324}, {104, 948}, {775, 890}, {107, 714}, {497, 122}, {1121, 280}, {433, 835}, {1113, 1476}, {1460, 1326}, {1425, 152}, {1102, 1354}, {349, 724}, {762, 1677}, {1588, 1221}, {1583, 1487}, {509, 1109}, {133, 940}, {1467, 546}, {790, 242}, {562, 28}, {1115, 1047}, {276, 1578}, {804, 55}, {1298, 315}, {1020, 161}, {1507, 704}, {995, 915}, {98, 1167}, {1040, 567}, {254, 365}, {1739, 1594}, {1144, 535}, {855, 1015}, {1570, 1466}, {34, 242}, {209, 552}, {908, 441}, {1137, 614}, {642, 154}, {51, 1361}, {1179, 886}, {151, 6}, {1494, 621}, {1095, 1668}, {1231, 327}, {1696, 1213}, {531, 838}, {625, 474}, {906, 333}, {1093, 1598}, {1404, 1681}, {1208, 1667}, {1496, 1652}, {551, 1453}, {227, 838}, {1675, 269}, {877, 217}, {1257, 909}, {1341, 1622}, {1052, 80}, {43, 361}, {498, 157}, {703, 9}, {448, 1133}, {1502, 1459}, {868, 881}, {1399, 835}, {746, 563}, {1694, 92}, {1667, 95}, {457, 1083}, {1750, 1495}, {1683, 824}, {3, 1751}, {"infinity","infinity"}}

附錄 5 執行結果

```
{{3, 2}},
 {{22115778, 21746894}},{{5348344, 12376088}},{{11520367, 4464963}},
 {{16221201, 10758683}},{{21577694, 20660989}},{{11410487, 5451470}},
 {{1960613, 7313588}},{{12598972, 21411210}},{{18200177, 2646115}},
 {{1193954, 2846940}}, {{19592287, 17407218}},{{3896312, 11617285}},
 {{20906199, 19516984}},{{13828997, 9129102}},{{12124523, 21341912}},
 {{22760508, 22261613}},{{11348512, 3974123}},{{6750045, 2872329}},
 {{8087735, 20792487}},{{22872464, 7466036}},{{16840040, 22255689}},
 {{6228475, 18851141}},{{17892848, 6938383}},{{9573150, 22962678}},
 {{22419648, 15846205}}, {{13070345, 16522034}},{{21770991, 9912795}},
 {{9607781, 2673456}}, {{19857970, 12341279}},{{6506173, 17405295}},
 {{6979688, 14918868}}, {{9247195, 36868}}, {{9023450, 681845}},
 {{14048608, 18686359}}, {{6480827, 17361553}}, {{16269570, 5001318}},
 {{9281377, 10264653}},{{3315084, 185886}}, {{1833292, 2119737}},
 {{23395128, 14781368}}, {{17819262, 22922359}}, {{21089047, 22861543}},
 {{2237638, 1019356}}, {{13642621, 10922053}}, {{9409161, 20106047}},
 {{11207632, 20159491}}, {{13489773, 9337811}}, {{16458894, 13752276}},
 {{21395431, 1229474}}, {{9436609, 2648423}}, {{11580304, 9739827}},
 {{354123, 2839729}}, {{23505898, 3922379}},{{1281956, 5347320}},
 {{7864762, 7192504}}, {{8981765, 18035018}}, {{2180893, 17354768}},
 {{4177837, 16274432}}, {{15969807, 18820217}}, {{1904589, 5472446}},
 {{1529545, 18950172}}, {{3354279, 23320401}},{{5289730, 8319496}},
 {{11852428, 2702995}}, {{21756996, 1823790}}, {{22360996, 23440594}},
 {{1825515, 11135210}}, {{2343827, 22684952}}, {{11953263, 9170209}},
```

{{16629396, 8698806}}, {{21366368, 21076374}}, {{16916611, 5437874}},
{{23457585, 9721616}}, {{4108237, 3220420}}, {{6515457, 6663419}},
{{13034330, 1922178}}, {{18955993, 16136310}}, {{2369086, 814475}},
{{10887203, 1549661}}, {{22416049, 6954571}}, {{19827044, 23409828}},
{{13468812, 5244306}}, {{3483649, 2336329}}, {{7027938, 18532784}},
{{15951269, 20855840}}, {{17384985, 11596515}}, {{23017100, 12574629}},
{{17573381, 11207195}}, {{5404227, 10202982}}, {{11611114, 18994670}},
{{3485754, 10902149}}, {{11010244, 7622116}}, {{22641669, 13539254}},
{{19256915, 7953001}}, {{18778828, 19736505}}, {{9534633, 9618090}},
{{10031280, 14206578}}, {{2229882, 14373979}}, {{5948621, 6486622}},
{{6936014, 6144419}}, {{7024784, 5477042}}, {{1450490, 16113504}},
{{7865291, 2495041}}, {{22954591, 1957942}}, {{4430024, 264082}},
{{12380408, 17587450}}, {{5720761, 19365790}},
{{"factor=", 1753}},
{{"factor=", 1753}},

參考資料

[1]沈淵源，密碼學之旅與 MATHEMATICA 同行，橢圓曲線與公鑰密碼系統，全華科技圖書出版，2006 年。

[2]沈淵源，不可能的任務-公鑰密碼傳奇，三民書局出版 2015 年。

[3]W.Diffie and M.E.Hellman, "New Directions in Cryptography" (1976)

[4] 周秉慧 楊中皇，植基於橢圓曲線的視窗圖形化介面大數分解工具之設計與實現— 以 GMP-ECM 為核心，2005 年。

[5] 陳榮傑，RSA 大數分解，2000 年。

[6] I. Blake; G. Seroussi; N. Smart. "Elliptic Curves in Cryptography." LMS Lecture Notes. (2000)

[7] Lenstra, A. K.; Lenstra Jr., H. W., eds. The development of the number field sieve. (1993)

[8] Lenstra Jr., H. W.. "Factoring integers with elliptic curves".(1987)