植基於橢圓曲線質數視窗化介面之質數判斷工具之設計與實現

DESIGN AND IMPLEMENTATION OF AN ECC-BASED PRIME NUNBER PROVING TOOL ON WINDOWS PLATFORMS

戴志坤

國立高雄師範大學

資訊教育研究所

tai.tomy@gmail.com

彭俊智

國立高雄師範大學

資訊教育研究所

borispong@gmail.com

楊中皇

國立高雄師範大學

資訊教育研究所

chyang@computer.org

摘要

於公開金鑰加密系統上面，數論扮演極重要的角色，尤其是質數的判斷與產生，更是研究的主題之一，例如在 **RSA** 中的 N 為兩個大質數 **P**、**Q** 兩個值的乘積，而 **P** 和 **Q** 的質數產生都必須用到質數的產生與判斷，當質數的值越大的時候，相對的破解的時間與複雜度因此而增加；現行質數判斷與產生的程式大部分都是在 **Linux** 平台上面執行或是以 **DOS** 平台來執行，使用者在使用上都必須要打入複雜的指令，有鑑於此，本研究將要整合各類質數判斷演算法，以 **Microsoft Windows XP** 為平台、**Borland C++ Builder 6.0** 為開發平台，設計一個以便利使用的視窗化介面，供選擇所需的演算法工具；此工具的演算法中尤以近年密碼學中的熱門的橢圓曲線所衍生之橢圓曲線質數判斷法（**Elliptic Curves Primality Proving**，**ECPP**）為重，利用橢圓曲線的特性，判斷所需的大質數。

關鍵字：公開金鑰密碼系統、橢圓曲線質數判斷法**(ECPP)**、橢圓曲線、**ECM**

1. 緒論

Diffie和**Hellman**在 **1976** 年提出公鑰密碼系統**(public-key cryptosystem)**後，許多演算法，例如：**RSA**、**Diffie-Hellman**密鑰交換、**ElGamal**、數位簽章（**DSS**）相繼被提出，應用在資料加密以及數位簽章等用途。這些公開金鑰密碼系統都有一個共通點，就是需要大質數來建構其系統，而且這些公鑰密碼系統的「安全強度」往往取決於所選擇的質數大小以及強弱，如**RSA**演算法需要兩個質數**p**和**q**相乘得到的公開金鑰**N**來當作系統運作時的參數，而強度決定於「大數分解」的困難度；**Diffie-Hellman**演算法以及數位簽章標準**(Digital Signature Standard)**則需要質數p來構成一個有限體**(finite field) Zp**以供系統運作，而對於**Diffie-Hellman**演算法及**DSS**而言，其強度則決定於離散對數的問題；**ElGamal**密碼系統利用質數來建立一個有限場**Zp**，產生公鑰及私鑰；橢圓曲線密碼系統利用質數在橢圓曲線方程式上可找到整數解的特性來產生其公鑰與私鑰。所以質數測試**(Primality Testing)**可有效地找出質數，提供公開金鑰系統來使用，因此如何找到一個有用的大質數在公開金鑰系統中是一個重要的課題。

2. 文獻探討

2.1 質數

質數或稱為素數，就是一個正整數，除了本身和 1 之外並沒有其他的因數。例如 **2**、**3**、**5**、**7**、**11** 等為質數，**4**、**6**、**8**、**9** 則不是，後者稱為合數。從這個觀點可將整數分為兩種，一種為質數（**primility**），一種為合數（**Composite**）。（有人認為數目字 **1** 不該稱為質數）高斯提出『唯一分解定理』學說，說明了任何一個整數可以寫成一串質數相乘的

積。例如 $28=7\times2^2$，$33=1\times33$，$84=2^2\times3\times7$，也就是說，任何數都由質數構成的。「質數有無限多個」，歐幾里德**(Euclid)**於西元前300年左右利用反證法輕易證明了。

證明：

假設質數有限個，共有 **n** 個，**p1、p2、p3、...、pn**，如果有一個數是 **P= p1×p2×p3×.....×pn+1**，因為 **p1、p2、p3、...、pn** 都不能整除 **P**，所以 **P** 的正因數只有 **1** 和 **P**，所以 **P** 一定是質數。而這結果顯然和假設不同，因此，質數是無限多個。

2.2 最大的質數

目前(**2005** 年 **12** 月)人類發現最大的質數為梅森質數（**Mersenne Prime**），共 **9,152,052** digits：

$2^{30402457}-1$（**9152052** 個數字）

此質數由美國密蘇里大學 **Dr. Curtis Cooper and Dr. Steven Boone** 使用以 **C** 語言為基礎的 **Glucas** 程式，花了約五天的時間測試出，其中 **Glucas** 程式以 **Lucas-Lehemer** 演算法為主。

$Let\ M_p=2^p-1$

$Let\ p$ 是一個質數

$if\ only\ if\ \ s(p-1)=0(\bmod M_p)$ 其中

$S_1=4$，$S_n=S_{n-1}{}^2-2$

則 M_p 是質數

圖 1 **Lucas-Lehemer** 演算法

2.3 質數的機率

數論上有個重要的函數 $\pi(x)$ 他代表所有不大於 **x** 質數的個數，例如。另外質數定理（**Prime Number Theorem**）告訴我們當 x 趨近無限大時，$\pi(x)$ 大約等於 $\frac{x}{\ln x}$，其中 lnx 代表 x 的自然對數值。其公式如下：

$$\lim_{x\to\infty}\frac{\pi(x)}{\frac{x}{\ln x}}=1 \qquad (1)$$

若隨機取 **150** 位數字，則質數的機率可用質數定理算出：

機率（**150** 位隨機整數為質數）

$$\frac{\pi(10^{150})-\pi(10^{149})}{10^{150}}\approx\frac{\frac{10^{150}}{150\times(\ln10)}-\frac{10^{149}}{149\times(\ln10)}}{10^{150}}= \qquad (2)$$

$$\frac{1}{150\times(\ln10)}-\frac{\frac{1}{10}}{149\times(\ln10)}=\frac{134}{22350\times(\ln10)}\approx\frac{1}{384}$$

可見位數越多，質數的密度會越來越小。

2.4 質數的測試

質數的測試是取決於正整數的分解，而質數都是奇數的觀念，只要找出能夠分解的質因數，就可以確定該數是否為質數。質數的測試法可分為**確定式質數測試法(Deterministic Primality Test)**和**機率式質數測試法(Probabilistic Primality Test)**，確定式質數判斷法所測試出來的質數我們稱為確定質數（**guaranteed Prime**），而機率式質數測試法所測試出來的質數我們稱為可能質數（**Probabilistic Prime**）或假質數（**pseudo-primes**），確定式質數測試法，經過保證測試出來的結果一定是質數，因為耗費的時間極長，比較適合應用在機密性較高的軍事、商業等方面，例如試除法、**ECPP**、**AKS**、**Lucas-Lehemer**；至於機率式質數測試法雖然結果不保證一定是質數，但都會要求極低的錯誤率，適合用在機密性不高的個人應用方面，例如費瑪測試法、米勒拉賓測試法，以下介紹目前常用的幾種質數測試的演算法。

2.4.1 費瑪定理（Fermat Theorem）

若 n 是質數，則對所有 **GCD（a,n）＝1**

$$a^{n-1}\equiv1\ (\bmod n) \quad (3)$$

實際上當 n 不是質數的時候，有可能存在這樣一個 a，使得 $a^{n-1}\equiv1\ (\bmod n)$，這時候我們稱 n 為基於 a 的偽質數**(base-a pseudoprime)**，這類質數可以使用費馬檢驗**(Fermat testing)**來檢驗，這個定理是根據費瑪定理的逆方向設計的質數檢驗法。

Step1. **for i = 1 to t do**

Step2. 隨機挑選 **a**，**2≤a≤n－2**

Step3. **$r = a^{n-1} \pmod n$**

Step4. 如果 **$r \neq 1$ then output**

“合數”，否則 **output** “質數”

圖 2 費馬檢驗

然而在實務上，會先嘗試用 **a＝2**，也就是說如果 **2≤a≤n－2** 且**$a^{n-1}\equiv 1 \pmod n$**，則**n**為質數，若不同餘，則**n**非質數。

另外有一類合數，滿足對所有與n互質的數**a**，**$a^{n-1}\equiv 1 \pmod n$**成立，我們稱為非質數的卡邁克爾數（**Carmichael Number**），利用費瑪定理無法剔除這些非質數。

2.4.2 米勒－拉賓質數判斷法（Miller-Rabin primality test）

實務上最常使用的質數檢驗法為米勒-拉賓**(Miller-Rabin)**檢驗，它的計算結果為非質數的判斷，其規則以大於 **2** 的質數都是奇數的觀念，隨機挑選一個奇數 **N** 來進行判斷。同樣的也挑選 **t** 次不同的 **a** 值進行檢查，使得產生偽質數的機率降低。米勒-拉賓提出的這種方法證明每挑選一次 a 值進行檢查，結果是偽質數的機率小於或等於 $\left(\frac{1}{4}\right)^k$，經過 k 次之後我們就可以把錯誤率降到達到可以接受的範圍之內，RSA 所能接受的誤差標準為 2^{-100}，所以大約反覆執行 50 次 $\left(\frac{1}{4}\right)^{50} \cong 2^{-100}$ 就可以達到標準的範圍。

奇數 N，N-1= $2^s \times t$，t 為奇數，判斷 **N** 是否為質數

Step1. 隨機挑選整數 **a**，**2≤a≤N－2**

Step2. 計算 **y=a^t mod N**

Step3. 如果 **y=1 or y=N-1**，則 **N** 可能為質數，結束

Step4. **For** $i \leftarrow 1$ **to s**

Step5. **do** y^2 **mod N** 如果 **y=N-1**，則到 Step6

如果 **y=1**，則 **N** 為非質數，結束

Step7. 如果 **y=N-1**，則 **N** 可能為質數，結束

Step8. 如果 **y≠N-1**，則 **N** 為非質數，結束

圖 3 米勒-拉賓檢驗

2.4.3 ECPP 質數判斷法

橢圓曲線的技術應用近年來在密碼學中，如數位簽章、金鑰交換及加解密，和大數分解 **(Factorization)** 與質數判斷 **(Primality testing)**上因為執行效率較其他確定質數判斷法效率高，漸被廣泛使用，本研究中所開發的程式就加入了橢圓曲線的質數判斷。

橢圓曲線的通用方程式如下：

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (4)$$

用於密碼學技術的橢圓曲線是由滿足上述方程式的所有點**(*x*, *y*)**及一個無限遠點**(point at infinity) *O*** 所形成的集合，其中座標*x*與*y*屬於某個有限體**(finite field)**。目前軟硬體具體實現的橢圓曲線有限體為質數體**(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。

橢圓曲線的加法運算：

如果是質數體，橢圓曲線方程式為

$$y_2 = x_3 + ax + b \qquad (5)$$

$\boldsymbol{P} = (x_1, y_1)$與$\boldsymbol{Q} = (x_2, y_2)$為橢圓曲線上的任意兩點，但$\boldsymbol{P} \neq \boldsymbol{O} \neq \boldsymbol{Q}$，則我們有下列兩點加法的運算規則：

1. $P + O = O + P = P$ (6)
2. $(x_1, y_1) + (x_1, -y_1) = P + (-P) = O$ (7)
3. $P + Q = R = (x_3, y_3)$ (8)

$$x_3 = \lambda_2 - x_1 - x_2 \qquad (9)$$

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

$$\lambda = \begin{cases} \frac{x_2 - x_1}{y_2 - y_1} if \quad p \neq Q \\ \frac{3 x_1^{\,2} + a}{2 y_1} if \quad p = Q \end{cases} \qquad (10)$$

如果是二元體，橢圓曲線方程式為

$$y^2 + xy = x^3 + a x^2 + b \qquad (11)$$

加法的規則改為 $P+Q=R=(x_3, y_3)$ (12)

$$x_3 = \begin{cases} \lambda^2+\lambda+x_1+x_2+a & if \quad P \neq Q \\ \lambda^2+\lambda+a & if \quad p=Q \end{cases} \qquad (13)$$

則 $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$ (14)

$$\lambda = \begin{cases} \frac{x_2 - x_1}{y_2 - y_1} if \quad p \neq Q \\ \frac{3 x_1^{\,2} + a}{2 y_1} if \quad p = Q \end{cases} \qquad (15)$$

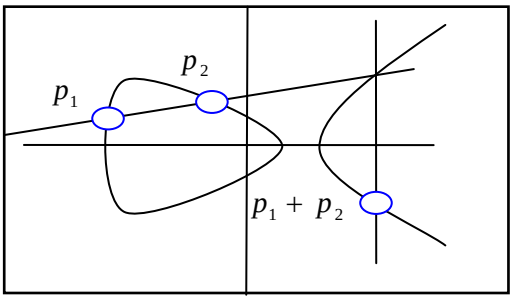


圖4　橢圓曲線幾何圖所表示兩點加法

ECPP演算法是由**H.W.Lentra**和**H.Cohen**所提出，並由**H.Cohen**和**A.K.Lentra**成功的完成執行，橢圓曲線測試法是利用質數可以在 $y^2 = x^3 + ax + b$ 橢圓方程式中找出 P_1、P_2、P_3、P_4 的整數解，而合成數不具有這樣的特性用來當做質數測試的方法，但計算效率仍沒有一般機率式質數測試法來得好，因此橢圓曲線測試法在實用上最常被當作機率式質數測試法的檢驗工具，用來確保其他機率式測試法結果的正確性。

假設 N and M = S×R（ S > 1）， R是質數

$R > (N^{\frac{1}{4}} + 1)^2$

如果在一個**non-singular**的橢圓曲線

$y^2 = x^3 + ax + b$ 對 N 模運算，其級數（order）為 M ，存在一個點 $P(x, y)$ 滿足

*P * M = Identity*

*P * S <> Identity*

則 N 是質數

圖5　**ECPP**演算法

2.4.4　AKS質數測試法

2002 年由印度的三位學者**Agrawal**、**Kayal**及**Saxena**所發表的**AKS**演算法，**AKS**的特色是利用數論裡的恆等式在有限場**(Finite Field)**中做多項式模運算，嘗試去代入所有不同的a和**r**檢查**n**是否符合下列式子的條件，將質數正確的判斷出來，這個演算法是一種確定式的質數測試法，也是多項式時間的演算法。

假設 a 和 n 互質

若n是質數

則滿足

$$(x-a)^n \equiv (x^n - a)(\bmod n, x^r - 1)$$

且對於所有的正整數 $a \leq 2\sqrt{r}\log_2(n)$

圖 6　AKS 演算法

表 1　質數判斷法的比較

演算法	特性	時間複雜度
Miller Rabin	**機率式**質數判斷法，也是使用最廣泛地演算法，屬於非多項式演算法	$O((\log n)^2)$
ECPP	**確定式**質數判斷法，使用橢圓曲線演算法	$O((\log n)^6)$
Lucas Lehemer	**確定式**質數判斷法，專門用於測試梅森質數	$(2^{n-2}\log(2+\sqrt{3}))$
AKS	**確定式**質數判斷法多項式演算法	$O((\log n)^{12})$

3. 系統分析

目前可以用來做質數判斷的函式庫有**NTL**、**LIDIA**、**GMP**，工具有

MATHEMATICA、**UBASIC**、**PRIMO** 等。(使用硬體配備 **CPU**：**Pentium Celeron 1.3GHz**、記憶體 **512 MB**

測試數據一（308digits）：
322638969161970869742926721592962852572876235477183189752231195515533726904774806425078297159311040039025083724608201473900514798177649413646622847127140417251504188523770133414476670262136197211023159073004450926935634808641348120991607495697378686487079903542524301817273543299305270754191090084623071（確定質數）**數據二**：$10^{480}+7$（**非質數**）
數據資料來源：http://www.ellipsa.net/

表 2：各類質數判斷軟體的比較

軟體名稱	操作環境	演算法	數據一 執行時間 數據二 執行時間
NTL	Linux	Miller-Rabin	＜1 秒
	Dos		＜1 秒
LIDIA	Linux	Miller-Rabin	約 3 秒
		ECPP	約 1 秒
MATHEMATICA	Windows	Miller-Rabin Lucas	約 2 秒
		ECPP	約 1 秒
UBASIC	Dos	APR	最大可測 $2^{455}-1$ 約 1 秒
GMP	Linux	Miller-Rabin	＜1 秒
	Dos		＜1 秒
Primo	windows	ECPP	19 秒
			約 1 秒

由以上結果可發現機率式和確定式的質數判斷法在非質數的判斷上效率差異不大，可能質數的判斷較有明顯的差異

3.1 MATHEMATICA

MATHEMATICA 是一套多功能整合性的數學軟體，其包含了質數判斷的 **primeQ** 模組，**primeQ** 結合了 **Miller-Rabin**、**Lucas** 和橢圓曲線的計算，雖然為視窗化介面之軟體，但仍要輸入指令才能進行判斷

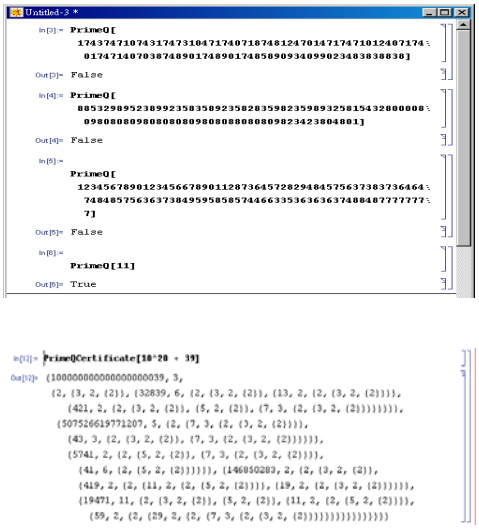


圖 7　**MATHEMATICA** 執行畫面

3.2　UBAISC

UBASIC是由一位日本大學(**Department of Mathematics, Rikkyo University in Tokyo**)木田佑司教授(**Prof. Yuji Kida van**)修改自**Basic**的一個程式語言，專門用於精密的數字運算，而且又內建了許多函式可供使用，其中質數判斷的方法是 **APR**（**Adleman-Pomerance-Rumely's primality test**)，改良自**Miller**質數判斷法。

圖 8　UBASIC 載入程式

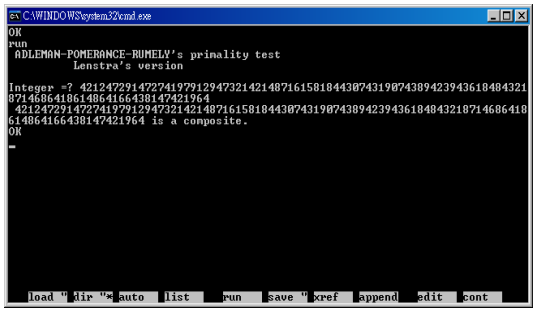


圖 9　UBASIC 執行結果

3.3　PRIMO

PRIMO 是一套由法國人 Colombes 以 ECPP 演算法為基礎的質數判斷工具，目前能夠判斷 7993 digits 的質數，但其效率較差，所利用的系統資源較大。另外必須使用特定類型的輸入檔才能進行判斷，造成使用上的不便。

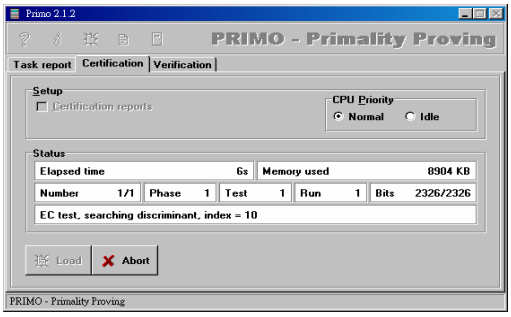


圖 10　PRIMO 執行畫面

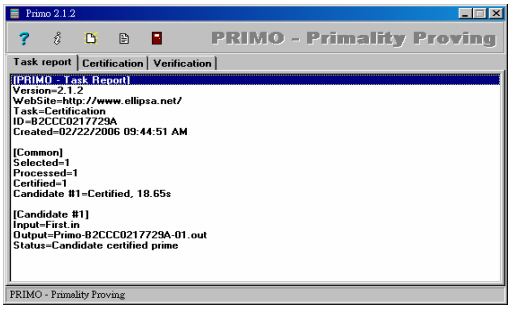


圖 11　PRIMO 執行結果

4. 系統設計 與實作

質數判斷與產生的程式大部分都是在 Linux 平台上面執行或是以 DOS 平台來執行，使用者在使用上都必須要打入指令才可以執行，有鑑於此，本研究將要利用各類演算法，以 Microsoft Windows XP 為平台、結合 Microsoft Visual C++ 6.0 與 Borland C++ Builder 6.0 為開發平台，將此工具以視窗化的介面來呈現，以便於使用。

4.1 函式庫移植

將各個軟體如 NTL、GMP、MIRACL、LIDIA、AKS、ECPP 等的原始碼移植 Windows 平台，方便日後編譯所需要的程式碼，透過 Visual C++將各原始碼，取其所要的 Source code 編譯成函式庫（LIB），而後為了要以視窗化的介面來呈現，且讓使用者可以輕易的點選所需的演算法來執行質數的判斷，或是產生。

因為編譯後的函式庫幾乎都是以 Visual C++來做的，無法直接在 Borland 中所使用，所以透過 Borland 所提供的一個轉換函式庫的方式將所有的函式庫 LiB 轉換成 Borland 可用的 LIB，並將運算結果顯示在 Memo 裡面。

4.2 程式結果展示

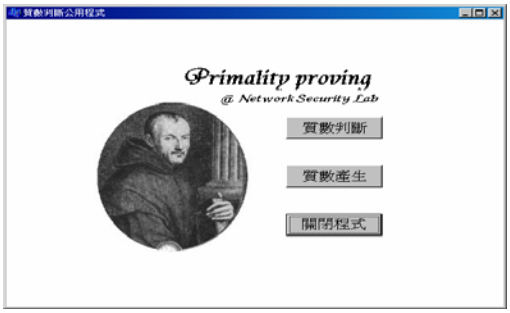


圖 12　質數判斷、質數產生程式選擇。

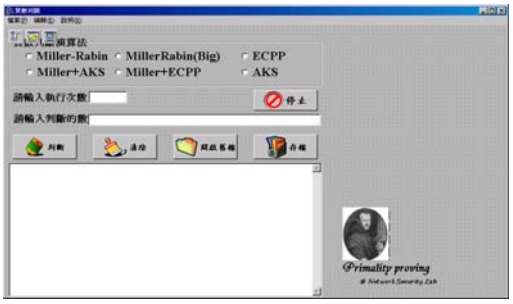


圖 13　質數判斷程式

圖 14　質數產生程式

4.3 軟體功能分析

質數判斷

4.3.1 Miller Rabin

使用由函式庫 **NTL**、**GMP** 所提供演算法，其判斷的函式速率相當，判斷 **500digits** 的以上的可能質數的時間均小於一秒。

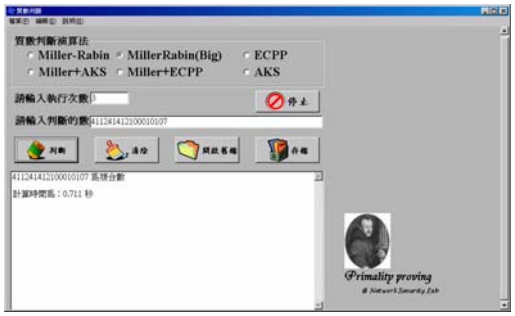


圖 15　**Miller Robin** 演算法判斷質數執行畫面

4.3.2 AKS

所執行的為三位印度學者所寫的質數判斷演算法，它所判斷出來的數為確定為質數，但效率上比其它的幾種演算法差，編譯時利用 **GMP** 函式庫所編譯的靜態函式庫 LIB 來增強其運算能力。

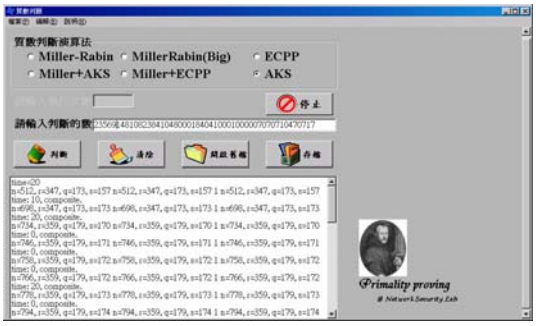


圖 16　**AKS** 執行判斷 **50digits** 數的畫面

4.3.3 ECPP

基於橢圓曲線加解密演算法上面所需要的質數判斷，演算法選擇由**F. Morain**在**Linux**系統中所寫的**ECPP** 質數判斷法，此演算法目前可判斷出大質數為*907^694+694^907*（**2578 decimal digits**），程式利用其 **Bignum**、**Bigmod**、**ECPP**三種模組編譯而成，另外嘗試加入函式庫**GMP**取代**Bignum**和**Bigmod**的模組，改善原本在**windows**系統下程式執行效率不佳的情形。

質數產生

4.3.4 NTL、GMP，MIRACL 質數產生

改良產生質數效率較高的三種函式庫的質數產生演算法，其方法是植基於 **Miller-Rabin** 演算法，可選擇所需 digits 或隨機產生大質數，其中以 **NTL** 的效能最高，可快速產生約 500digits 以上的可能大質數，另外兩種演算法因所產生的質數正確性較高，所以一並採用。

圖 17　質數產生執行結果

4.3.5 軟體使用

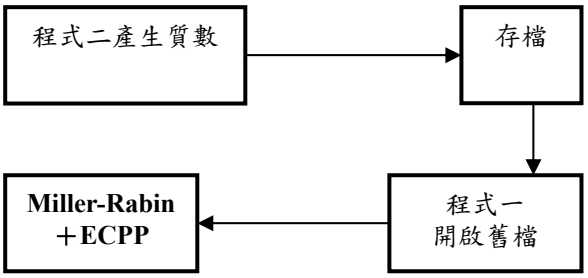
在質數判斷上面，我們提供了 **Miller Robin**、**AKS**、**ECPP** 以及實務上常用的接力執行的方式 **Miller+ECPP** 及 **Miller+AKS**，輸入方式可直接輸入，也可利用開啟舊檔（***.TXT**）的方式來進行輸入，在 **Miller-Rabin** 演算法的部份可以輸入執行次數，判斷完成後可將執行結果儲存。

圖 18　質數判斷功能模組

在質數產生方面提供了三種模式，分別利用數論函式庫中三個執行效率高的 **NTL**、**GMP** 及 **Miracl** 來實做，使用者只要 **Click** 開始產生鍵，就可隨機產生大質數。

圖 19　質數產生功能模組

在使用上建議按照以下模式可較有效率的獲得需要的大質數（以下質數判斷程式為程式一，質數產生程式為程式二）



4.3.6 效率分析

效率分析的測試使用軟、硬體配備為（CPU：Intel 1.8GHz、記憶體：DDR2 1024MB、OS：Windows XP professional SP2）

圖 20 測試數據是使用以 NTL 函式庫為基礎的質數產生器，在 2048bits 之前的時間均在 100 秒以內，但在 2048bits 以上的數時，所需時間遽增。

本研究於"產生質數"的程式上提供了三種改良函示庫 NTL、GMP 和 MIRACL 的質數產生方法，但在產生上面各都以不同的方式來呈現。

A. NTL 可以指定範圍，並產生一個大量的質數。
B. GMP 雖然在大數方面提供了非常快速的演算法，此處它也是給定一個範圍，產生的方式為此範圍內所有的質數。
C. MIRACL 的產生方式比較特別，必須分別給定 Spins 值 4digits 和 Seed 值 9digits，且產生的質數也不夠大。

當然 GMP 與 MIRACL 有其優點是 NTL 無法完成的，如給定範圍內的所有質數；為了要達到我們所要的目的（產生非常大的質數，如 512~4096bits），本文因此採用了 NTL 所提供的函示庫來呈現。

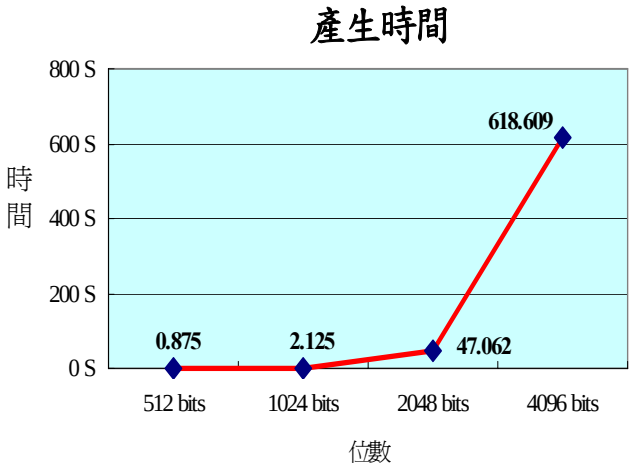


圖 20 NTL 質數產生時間分析表

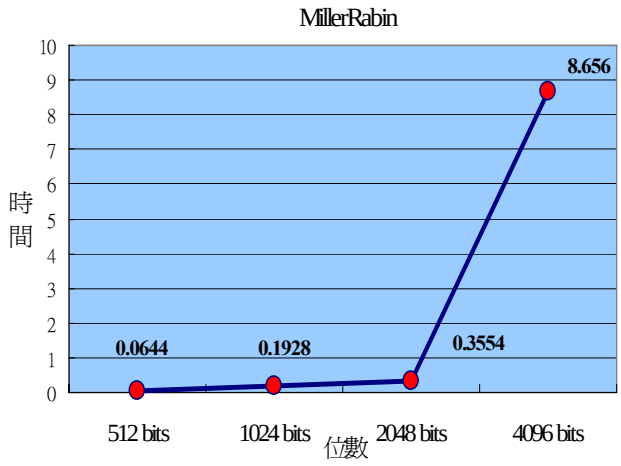


圖 21 Miller 質數判斷時間分析表

圖 21 測試數據是使用 GMP 函式庫的 Miller Rabin 演算法執行 50 次，判斷質數的時間隨著位數的增加，尤其是從 2048bits 以上變化最為明顯。

在"判斷質數"的程式上提供了三種演算法，因 AKS 執行效率差，故以實務上運用較為廣泛的 MillerRabin、ECPP 來展示。

圖 21 的 MillerRabin 質數測試為 GMP 函示庫所提供的，它在判斷質數的速度經測試雖比 NTL 提供的 Miller-Rabin 函示庫來得慢，但是較適合在大數上面的判斷，而 NTL 的函示庫雖在速度上面佔了許多優勢，但是可以判斷的數卻小於 GMP 的函示庫。

圖 22 為 ECPP 質數判斷法的時間分析，其效率較差，判斷時間約為 Miller-Rabin 的 1000 倍。

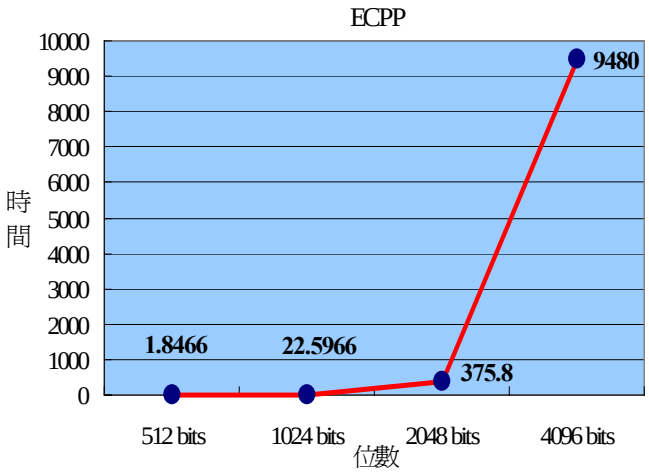


圖 22 ECPP 質數判斷時間分析表

單位時間：（秒）

表 3 與 Linux 平台質數判斷程式效率比較表

演算法	執行次數	平均執行時間 Windows 平台	相較於 Linux 平台
ECPP	1	4.3 秒	較差
ECPP +GMP	1	3.6 秒	相當
Miller Rabin	1	＜1 秒	相當
	10	＜1 秒	相當
	50	＜3 秒	相當
3226389691619708697429267215929628525728762354771831897522311955155337269047748064250782971593110400390250837246082014739005147981776494136466228471271404172515041885237701334144766702621361972110231590730044509269356348086413481209916074956973786864870799035425243018172735432993052707541910900846230 71 測試數據：資料來源(http://www.ellipsa.net/)			

5.結論

大質數在公開金鑰密碼學中重要不言而喻，要如何選擇適當的大質數作為加解密的金鑰，防止加密的檔案被輕易的破解，的確是一個重要的課題，然而判斷速度快的機率式的質數判斷法並沒有辦法完全確定質數的正確性，利用其加密的檔案被破解的機率相對提高，而效判斷速度較慢確定式的質數判斷法又沒有辦法有效率的提供大質數。

橢圓曲線的運用除了加解密，還可用來做質數的判斷，本研究所開發的系統集合了機率式質數判斷法的優點，並結合橢圓曲線的確定式質數判斷法的新技術，以 **windows** 的介面為基礎，解決使用者在 **Linux** 或 **Dos** 上需要輸入複雜指令的困擾，並提供更多的演算法來提供公開金鑰加解密所需金鑰之研究。

6.參考文獻

[1] 楊中皇，橢圓曲線密碼系統軟體實現技術之探討，資訊安全通訊，第十一卷第一期，頁 **15~25，2005** 年 **1** 月。

[2] 數學知識 EpisteMath (c) 2000 中央研究院數學所、台大數學系 http://episte.math.ntu.edu.tw

[3] 蔡孟凱、雷穎傑、黃昭維、陳錦輝、陳正凱(2003) ."C++ Builder 6 完全攻略”，金禾資訊。

[4] The Prime Page，http://primes.utm.edu/

[5] The ECPP homepage http://primes.utm.edu/

[6] Elliptic curves and primality proving, A. O. L. Atkin and F. Morain, Math. Comp. 61, 203, july 1993

[7] NTL: A Library for doing Number Theory http://www.shoup.net/ntl/

[8] Multiprecision Integer and Rational Arithmetic C/C++ Libraryhttp://indigo.ie/~mscott/

[9] A C++ Library For Computational Number Theory, http://www.informatik.tu-darmstadt.de/TI/LiDIA/

[10] R. P. Brent, "Primality testing," seminar presented at Clemson University, South Carolina,April 4 ,2003。

[11] F.Morain,”Easy number for the Elliptic Curve Primality Proving Algorithm” F. Morain, in Paul S. Wang, editor, ISSAC '92, pages 263－268.

[12] GNU Multiple Precision Arithmetic Library, http://swox.com/gmp/

[13] Shamus Software Ltd – MIRACL http://indigo.ie/~mscott/

[14] Ellipsa Homepage http://www.ellipsa.net/

[15] GIMPS Home Page http://www.mersenne.org

[16] Agrawal, N. Kayal, and N. Saxena, Primes in p http://www.cs.miami.edu/~burt/manuscripts/primes-in-p/primes_in_p_notes.pdf ,August 2002

[17] R. Bhattacharjee, and P. Pandey, "Primality testing "http://www.cse.iitk.ac.in/research/btp2001/primality.ps.gz ,IIT Kanpur, 2001.

[18] K. H. Rosen, Elementary Number Theory and Its Application, Addison-Wesley, 4th Edition,2000.

[19] A. Menezes, P. van Oorschot, and S. anstone, Handbook of Applied Cryptography, CRC Press, 3rd Edition,

1996.

[20] “Modern Primality Tests and therawal-Kayal-SaxenaAlgorithm”,http://wonka.hampshire.edu/~jason/math/thesis/div3.pdf,25th April 2003

[21] T. ElGamal, "Apublic key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. Inform. Theory, Vol.31, 1985, pp.469-472

[22] Implementation of the AKS Algorithm http://perso.wanadoo.fr/yves.gallot/src/