植基於橢圓曲線視窗化之大數分解暨質數判斷工具設計與實現

彭俊智 研究生　　楊中皇 教授

國立高雄師範大學 資訊教育研究所

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、產生以及大數分解，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；另一方面在大數分解上，若要分解 **RSA** 中 N 的因數卻是難以執行。現行大數分解以及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演算法，以 **Microsoft Windows XP** 為平台、**Borland C++ Builder 6.0** 為開發平台，輔以 **Microsoft Visual C++ 6.0** 為核心編譯，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**) 和大數分解 **GMP-ECM**、**NFS** 為主，最後也將 **CA** 整合進來，整個系統中包含了數論相關演算法。

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

1.緒論

加密演算法，可以分為對稱式加密與非對稱式加密，而 **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**）、**CA** 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且難以被分解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，就是一個正整數或稱為素數，除了本身和 1 之外並沒有其他的因數。舉個例子 **3**、**5**、**7**、**11**、**13** 等為質數，**4**、**6**、**8**、**9** 則不是，稱為合數或是複合數。從這個觀點可將整數分為兩種，一種為質數（**Primality**），一種為複合數（**Composite**）。有人認為數目字 **1** 不該稱為質數，因此高斯提出『唯一分解定理』Unique Decomposition Theorem 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，$77 = 1 \times 77$，$84 = 2^2 \times 3 \times 7$，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：**http://members.aol.com/DrMWEcker/Mersenne.htm**

$2^{30402457} - 1$（**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。

此方法假設 $2^q - 1$ 為質數，給定一個 q 值（odd）積數，來使用 Lucas-Lehmer 測試：

```
Lucas-Lehmer-Test（q）
    u := 4
    for i := 3 to q do
        u := (u^2 − 2) mod (2^q − 1)
    end do
    if u == 0 then
        2^q − 1 is prime
    else
        2^q − 1 is composite
    End if
EndTest
```

圖 1 Lucas-Lehemer 演算法 [10]

2.3 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 費瑪定理（Fermat Theorem）

若 n 是質數，則對所有 **GCD（a,n）＝1**

$$a^{n-1} \equiv 1 \pmod{n} \quad (1)$$

實際上當 n 不是質數的時候，有可能存在這樣一個 a，使得 $a^{n-1} \equiv 1 \pmod{n}$，這時候我們稱 n 為基於 a 的偽質數**(Base-A Pseudo prime)**，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**，$2 \le a \le n-2$

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

Step4. 如果 $r \neq 1$ **then output**

"合數"，否則 **output** "質數"

圖 2 費瑪檢驗 [1]

然而在實務上，會先嘗試用 **a＝2**，也就是說如果 $2 \le a \le n-2$ 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。

2.5 米勒－拉賓質數判斷法（Miller-Rabin Primality Test）

2.5 Miller-Rabin

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 次就可以達到標準的 $\left(\frac{1}{4}\right)^{50} \cong 2^{-100}$ 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 Miller Rabin [1]

2.6 ECPP質數判斷演算法

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：

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

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

$$y_2 = x_3 + ax + b \qquad (3)$$

$\boldsymbol{P} = (x_1, y_1)$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. $P + O = O + P = P$ (4)
2. $(x_1, y_1) + (x_1, -y_1) = P + (-P) = O$ (5)
3. $P + Q = R = (x_3, y_3)$ (6)

$$x_3 = \lambda_2 - x_1 - x_2 \qquad (7)$$

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

$$\lambda = \begin{cases} \frac{x_2 - x_1}{y_2 - y_1} if \quad p \neq Q \\ \frac{3x_1^2 + a}{2y_1} if \quad p = Q \end{cases} \qquad (8)$$

如果是二元體，橢圓曲線方程式為

$$y^2 + xy = x^3 + ax^2 + b \qquad (9)$$

加法的規則改為 $P + Q = R = (x_3, y_3)$ (10)

$$x_3 = \begin{cases} \lambda^2 + \lambda + x_1 + x_2 + a & if \quad P \neq Q \\ \lambda^2 + \lambda + a & if \quad p = Q \end{cases} \qquad (11)$$

則 $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$ (12)

$$\lambda = \begin{cases} \frac{x_2 - x_1}{y_2 - y_1} if \quad p \neq Q \\ \frac{3x_1^2 + a}{2y_1} if \quad p = Q \end{cases} \qquad (13)$$

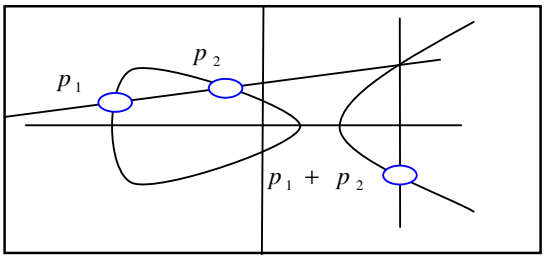

圖 4 橢圓曲線幾何圖所表示兩點加法 [2]

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。

Procedure ECPP（N）
（N is probably prime）
Step 1：set i = 0，N_0 = N；
Step 2：building the sequence：
While $N_i > N_{small}$

1. find a fundamental discriminant D_i which is good for N_i；in other words $N_i = \pi_i \pi_i'$ in $K = Q\left(\sqrt{-D_i}\right)$；
2. if one of the $W(-D_i)$ numbers $m_1, \cdots\cdots m_w$（$m_r = N_k(v_r \pi_i - 1)$ where v_r is a unit in K）is probably factored go to step 2.3 else go to 2.1；
3. store $\{i, N_i, D_i, v_r, \pi_i, m_r, F_i\}$where $m_r = F_i N_{i+1}$ Here F_i is a completely factored integer and N_{i+1} a problbly prime；set i = i+1 and go to step 2.1；

Step 3：Proving
for i = k down to 0

1. compute a root j of $H_{D_i}(X) \equiv 0 \bmod N_i$；
2. compute an equation of the curve E_i of invariant j and whose cardinality modulo N_i is m_i；
3. find a point P_i on the curve of E_i；
4. check the conditions of the theorem with $S = N_{i+1}$ and $m = m_i$ in other words check that $Q_i = F_i P_i \neq O_{E_i}$ but $sQ_i = O_{E_i}$；

Step 4：End（Finding m_r which is probably factored will be refered to as "finding a suitable m"）

圖 5 ECPP 演算法 [4]

2.7 AKS 質數判斷演算法

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。

Input: Integer $n > 1$
if (n is has the form a^b with $b > 1$) then output COMPOSITE
$r := 2$
while ($r < n$)
{
  if (gcd(n,r) is not 1) then output COMPOSITE
  if (r is prime greater than 2) then
    { let q be the largest factor of r-1
    if ($q > 4\text{sqrt}(r)\log n$) and ($n^{(r-1)/q}$ is not 1 (mod r)) then break
    }
  $r := r+1$
}
for a = 1 to 2sqrt(r)log n
{
  if ($(x-a)^n$ is not (x^n-a) (mod x^r-1,n)) then output COMPOSITE
}
output PRIME

圖 6 AKS 演算法 [6]

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	$(2^{n-2}\log(2+\text{sqrt}(3)))$
AKS	**確定式**質數判斷法，多項式演算法	$O((\log n)^{12})$
出處：Lucas [10]、ECPP, AKS [15]、Miller [11]		

2.8 大數分解

RSA 加密演算法是由 Ron Rivest、Adi Shamir、Len Adleman 三位學者於 1977 年所提出的。其公開金鑰中的[e,n]，其中 n 為兩個大數 p、q 的乘積，因此要破解密文的話就必須先將 n 分解開來;在過去的電腦等級幾乎是很難來做分解的動作。如今拜在科技進步，且加密的 bit 數都是 1024 bits 以上，這些只是時間的長短問題，不再是難題了。目前常用的大數分解演算有幾種常見的，像是試除法（Trial Division）、Pollard's P-1、Pollard's Rho、Williams P+1、ECM，MPQS、NFS 等等。

2.9 試除法（Trial Division）

試除法是大數分解演算法中最簡單容易理解的演算法。給定一個複合數 n，且看成是用小於等於 $\sqrt{n}$ 的每個質數去試除待分解的整數。如果找到一個數能夠整除盡，這個數就是待分解整數的因數。

試除法一定能夠找到 n 的因數，如果找不到其它的因數則這個演算法「失敗」，也就證明了 n 是個質數。某種意義上說，試除法是個效率非常低的演算法，如果從 2 開始，一直算到 $\sqrt{n}$ 需要 $\pi(\sqrt{n})$ 次試除，值得注意的是，對於隨機的 n，2 是其因數的機率是 50%，3 是 33%，等等，88% 的正整數有小於 100 的因數，91%的有小於 1000 **[18]**。

2.10 Pollard's P-1

Pollard's P-1 大數分解演算法是由 J.M.Pollard 於 1970 年提出的。其基本演算法的核心在於探討在所有 Z_p 群裡的元素，找出 N 的因素 p，演算法步驟如下：

1. 選擇演算法 B 的範圍，通常介於 10^5~10^6。
2. 計算 $m = \prod_{\substack{pprime \\ 1\le p\le B}} p^{\lfloor \log B/\log p \rfloor}$
3. 在 1 到 n 之間選擇一個隨機的正整數
4. 計算 d = gcd(a,n)
   假設 d = 1，go to 5;
   假設 d ≠ 1，回傳 d（找到因數 n）
5. 計算 a^m mod n;
6. 計算 e = gcd(a^m - 1,n);
   假設 e = 1，go to 1 且增加 B
   假設 e = n，go to 3 且修改 a
   假設 e ≠ 1 & e ≠ n，回傳 d（找到因數 n）

圖 7 Pollard's P-1 演算法（[5]第四頁）

2.11 Pollard's Rho

主要是基於主要是基於特別的大數分解演算法，於 1975 年由 John Pollard 所開發出來的，它有效於將一個複合數分解成小的因數。其理論基礎是基於 **Floyd's cycle-finding** 演算法；它會在一個隨機的方程式中找出 x and y 同餘 mod p，假設 p 是 n 的因數，且滿足 $1 < \gcd(|x - y|, n) \le n$。

輸入： n 為想要分解的數；$f(x)$，為隨機生成函數 mod n
輸出： n 的因數，或者錯誤（無法分解）。

1. $x = 2, y = 2; d = 1$
2. While $d = 1$:
   A. $x \leftarrow f(x)$
   B. $y \leftarrow f(f(y))$
   C. $d \leftarrow \text{GCD}(|x - y|, n)$
   D. If $1 < d < n$, then return d.
   E. If $d = n$, return failure.

圖 8 Pollard's Rho 演算法 [18]

2.12 Williams P+1

是一個類似於 Pollard's P-1 的大數分解演算法，由 H.C.Williams 所開發出來的。使用 Lucas Sequence 來做計算，假設 n 為被分解的數，且包含一個或是多個質因數 p 為 Smooth 數，則可以快速的找出因數 **[18]**。

2.13 MPQS

Quadratic Sieve 於 1981 年由 Carl Pomerance 學者所提出的演算法，而 MPQS（Multiple Polynomials Quadratic Sieve）為 QS 的變形。假設 n 為想要分解的數，而 QS 企圖想要找出兩個數 x，y 且滿足 $x^2 \equiv y^2 \bmod n$，相當於 $(x-y)(x+y) \equiv 0 \pmod{n}$ 然後再使用 Euclidean 演算法來計算 $(x-y, n)$。而 MPQS 則是將原本的 Q(x) 方程式改成以多項式的方式即為 $Q(x) = (x + \lfloor \sqrt{n} \rfloor)^2 - n \Rightarrow Q(x) = ax^2 + 2bx + c$ 此時假設 a 為平方，$0 \le b < a$ 且 $b^2 \equiv n \mod a$ **[7]**。

2.14 ECM

Elliptic Curve Method 是由 H. W. Lenstra 於 1987 所發表基於橢圓曲線的大數分解演算法，其方法是由先前的 Pollard's P-1 的 Multiplicative Group（乘法群）演變成以隨機的一個橢圓曲線上的點的群。其分解的複雜度取決於其因數的大小，跟原本分解的數大小無關。

n 為一個複合數，且為想要分解的數，其必須 $n \ge 2$。

1. 測試 $GCD(n,6) = 1$，其 n 不為某數的平方。例如，$n \ne m^r$，其 $r \ge 2$。
2. 選擇一個隨機曲線 $E = E_{a,b}$，和一個隨機點 $P = (x, y)$ 在 E 上。選擇隨機整數 $a, x, y \in [0, n-1]$，
   $b = y^3 - x^3 - ax$，再測試
   $g = GCD(4a^3 + 27b^2, n) = 1$
   假設 g = n 則到第二步。
   假設 1<g<n，則回傳 g。
3. 選擇一個整數 k 可以被小質數的平方整除 $(\le B)$，$k = \prod_{l \le B} l^{\alpha l}$。當 l 為質數像是 $l^{\alpha l} \le C$ 時 $\alpha l = \lfloor \log C / \log l \rfloor$ 為最大的指數。
4. 計算 kP mod n。
5. 假設第四步出現錯誤，則因為斜率 slope mod n 不存在，因此對於 n 來說 $x_1 - x_2$ 或者 $2y_1$ 都不是質數，最後找出 n 的因數。假設第四步成功，則回到第一步。

圖 9 ECM 演算法 [17]

2.15 NFS

在數學理論中，新興的 NFS（Number Field Sieve）演算法對於分解大於 100 位以上的數是很有效率的，主要是 Quadratic Polynomial（二次多項式）選擇上的改進。起初是為了要以特別的方式 x^3+k 來快速的分解大數，因此於 1990 年的時候 SNFS（Special Number Field Sieve）演算法被用來分解特定的 Fermat Number $r^e \pm s$，r、e、s∈ Z，且 e > 0，其分解的速度快，分解的位數也可高於 155 digits 以上，但因為只能夠使用在特定的數，所以後來改採用 GNFS（General Number Field Sieve），實際上 GNFS 也比較常用於數論的研究上面。GNFS 的演算法主要分成以下幾個步驟： **[12]**

A. Selecting Polynomial
B. The Relational Factor Base
C. The Algebraic Factor Base
D. The Quadratic Character Base
E. Sieving
F. Forming the Matrix
G. Finding Dependencies
H. Computing the Explicit Square Root in Z[θ]
I. Determining Applicable Finite Field
J. Square Root in Finite Field
K. Using the CRT（中國剩餘定理）
L. Putting it Together

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。

大數分解的函式庫有 **GMP_ECM**、**MIRACL**、**MSIEVE**、**Dixon's**，工具有 **UBASIC**、**SAI Factorizer** 等等(使用硬體配備 **CPU**：**Pentium Centrino 1.83GHz**、記憶體 DDR2 **1024 MB**)

測試數據一（**308 digits**）：**322638969161970869742926721592962852572876235477183189752231195515533726904774806425078297159311040039025083724608201473900514798177649413646622847127140417251504188523770133414476670262136197211023159073004450926935634808641348120991607495697378686487079903542524301817273543299305270754191090084623071**（確定質數）**數據資料來源**：http://www.ellipsa.net/

表 2 質數判斷演算法時間效能分析

(測試數據一)

原始碼/軟體	操作平台	演算法	執行時間(秒)
NTL	**Linux Dos**	**Miller-Rabin**	＜1 秒
LIDIA	**Linux**	**Miller-Rabin ECPP**	約 3 秒
MATHEMATICA	**Windows**	**Miller-Rabin Lucas ECPP**	約 2 秒
UBASIC	**Dos**	**APR**	最大可測 $2^{455}-1$ 約 1 秒
GMP	**Linux Dos**	**Miller-Rabin**	＜1 秒
Primo	**windows**	**ECPP**	19 秒

表 3 大數分解演算法時間效能分析一

測試數據二（**40 digits**）：**1977638319177019201778121983683193287949**

原始碼/軟體	操作平台	演算法	執行時間(秒)
MIRACL	Linux/DOS	Williams P+1	0.375
	Linux/DOS	Pollard's P-1	0.125
	Linux/DOS	ECM	0.125
GMP_ECM	Linux/DOS	ECM	0.812
MSIEVE	DOS	GNFS	0.391
Factorizer	DOS	MPQS	2.64

表 4 大數分解演算法時間效能分析二

測試數據三（**60 digits**）：**864348107145237357955506220129933199636770452842930532847781**

原始碼/軟體	操作平台	演算法	執行時間(秒)
MIRACL	Linux/DOS	Williams P+1	0.328
	Linux/DOS	Pollard's P-1	0.118
	Linux/DOS	ECM	0.171
GMP_ECM	Linux/DOS	ECM	1.218
MSIEVE	DOS	GNFS	0.609
Factorizer	DOS	MPQS	6.7915

註 1：表 3、表 4 的 GMP ECM 於執行的時候如果參數值給比較小，則執行速度越快，相對正確率降低。

註 2：表 3、表 4 之執行時間為實際五次分解大數的平均時間。

3.1 SAI Factorizer

此套軟體是由 Hermetic Systems 公司的 Peter Meyer 針對大數處理所開發的商業付費軟體，且可以應用在 Windows（DOS）模式下執行，而此套軟體主要的功能可以分為三種。第一種採用（Trial Division）試除法、其二採用 Pollard's Rho 演算法、最後一種為採用 MPQS 演算法。

試除法：其限制為最多可以七位數，而分解的速度也可在 1 秒內完成。

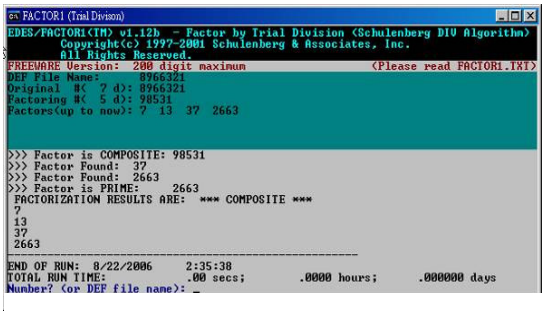

圖 10 SAI Factorizer Trial Division

Pollard's Rho 演算法：最大的限制為 200 位數，可是在時間成本上會花比較多。在進行此演算法之前，會先針對輸入的值以 Miller Rabin 演算法做測試，判斷是否為質數，如果為質數則不需再分解。

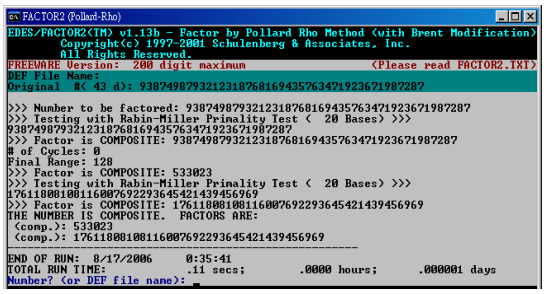

圖 11 SAI Factorizer Pollard's Rho

MPQS 演算法：最大的限制為 100 位數，於速度上面，遠快於 Pollard's Rho 演算法，同樣的數字 40 digits，採用 MPQS 可以於 2.7 秒內完成，但是如果採用 Rho 演算法可能得要花上四到五個小時時間，而有時候不一定會成功的分解出來。

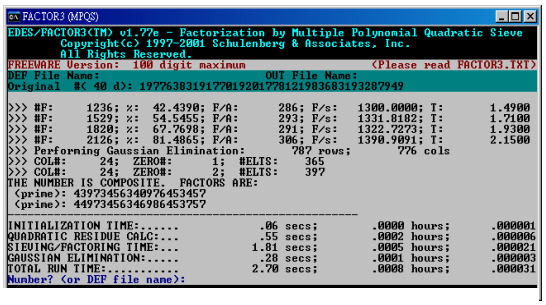

圖 12 SAI Factorizer MPQS

3.2 MIRACL

是由 Shamus Software Ltd 針對大數處理以及密碼學所開發的函示庫，且是提供了 Windows(DOS)、Linux 平台完整的 Source Code 軟體，目前版本為 5.10 版。例如 RSA、Diffie-Hellman Key Exchange、DSA、ECC 等等。在此我們主要是使用大數處理部分的函示庫，例如 Trial Division、Pollard's P-1、Pollard's Rho、Williams P+1、ECM 等大數分解演算法，於速度上，整體上都還有不錯的表現，但是在分解的位數上最好不要超過 100 位數，而官方網站則是建議在 80 位數之內，可以比較準確的分解。

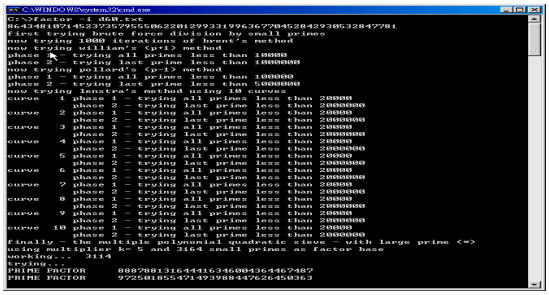

圖 13 MIRACL 大數分解畫面（DOS）

3.3 GMP_ECM

此 Source Code 是由 ECMNET 計畫組織所開發的程式碼，主要是針對使用橢圓曲線的方式來進行大數的分解，本身的程式碼也是基於 GMP(GNU Multiple Precision Arithmetic Library）大數處理的函示庫與 ECM（Elliptic Curve Method）演算法來結合，加快本身的計算上的速度。GMP ECM 原本的應用環境為 Linux 平台，如果要使用在 Windows 平台則必須透過模擬 Linux 環境的軟體 Cygwin 等等。而本研究則是透過 Visual C++ 6.0 將此順利的移植到 Windows 平台上，如圖 14 顯示於 DOS 下。

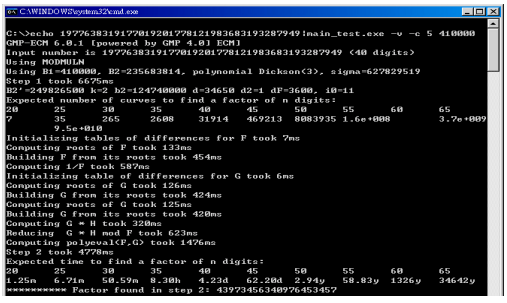

圖 14 GMP ECM 大數分解畫面（DOS）

3.4 Msieve

此 Source Code 是由 Jasonp 所開發的程式，目前最新的版本 1.07，提供了 Linux 以及 Windows 兩大平台的版本，演算法上主要是針對 NFS，在位數上的分解，最好控制在 100 位以內。參數的選擇上，可以針對 NFS 的 Polynomial Selection 或是 Sieving 部分，以及結合兩種來進行。

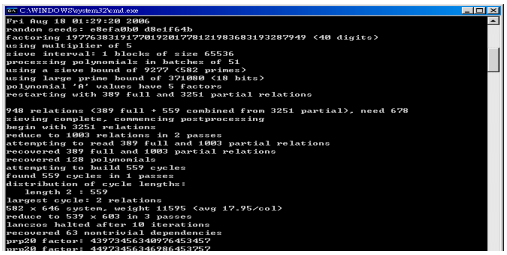

圖 15 Msieve 對 NFS 之大數分解畫面

3.5 CA（憑證管理）

在開放式網路上，使用公開金鑰基礎建設 PKI 作為資訊驗證及數位資料安全交換的文件。憑

證是由核發憑證的主管單位以數位方式簽署,可以發給使用者、電腦或服務,且安全地將公開金鑰連結到擁有相對私密金鑰的實體。目前最廣泛的憑證格式為 X.509。

本研究為了要分解 CA 中所使用的 Public Key 因而使用一個 CA client [2]端程式，後端則是使用 Open CA 於 Linux 上架設 CA Server，之後透過 CA client 端程式申請一個憑證，並通過 RA 驗證之後，由自行開發的另一個程式模組將 CA 匯入，擷取 Public Key 部分，經解碼後，以十進位模式呈現，最後再匯入大數分解程式來進行分解的動作。

4 系統實作

質數判斷與大數分解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。

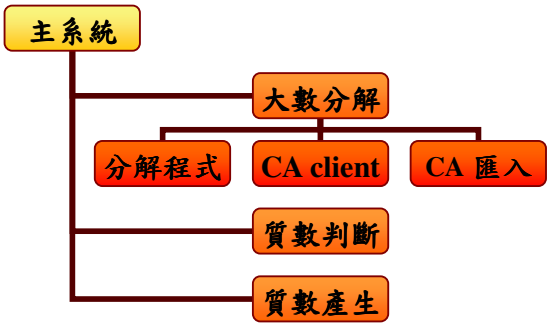

圖 16 系統架構圖

4.1 函式庫移植

將各個軟體如 NTL、GMP、MIRACL、AKS、ECPP 、GMP-ECM 、Msieve 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

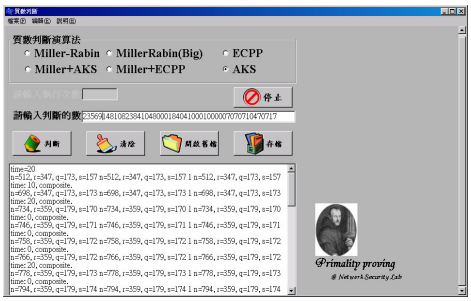

圖 17 質數判斷程式

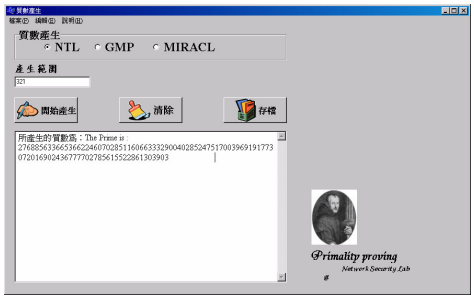

圖 18 質數產生程式

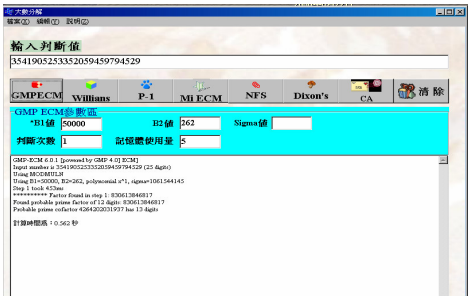

圖 19 大數分解程式

圖 20 CA client 程式

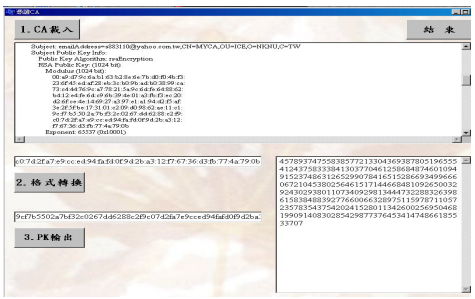

圖 21 CA Public Key 匯入程式

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。

為了要驗證所產生的大數或是大質數，則必須仰賴著大數分解的演算法，如果此大數位數大，則增加其破解的時間複雜度。

橢圓曲線的運用除了加解密，還可用來做質數的判斷及大數分解上，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，以及基於橢圓曲線之大數分解，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，”網路安全理論與實務”， 金禾出版社。ISBN：9861491767

[2] 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，p.15~25，2005 年 1 月。

[3] 鄭佩技、楊中皇，”CA Live-CD:開放原始碼下中文化認證中心的設計與實現”，第十六屆國際資訊管理學術研討會，2005 年 5 月。p.3~10

[4] A. O. L. Atkin and F. Morain，”Elliptic Curves and Primality Proving” ， 1992 http://www.informatik.uni-bonn.de/~adrian/ecpp/1992-atkin-morain-elliptic.pdf，p.9~10。

[5] Anne-Sophie Charest ， ”Pollard’s P-1 and Lestra’s Factoring Alghrithm” ， October/2/2005。 http://www.math.mcgill.ca/~darmon/courses/usra/charest.pdf，p.3~p10

[6] Andreas Klappenecker，”An Introduction to the AKS Primality Test” ， Sep/4/2002 ， http://faculty.cs.tamu.edu/klappi/629/aks.pdf ，p.1~2。

[7] Eric Landquist ， ”The Quadratic Sieve Factoring Algorithm” ， Math 448 ： Cryptographic Algorithm ， Dec/14/2001 。 http://www.math.uiuc.edu/~landquis/quadsieve.pdf，p.2~6。

[8] Eli Biham ， ”Factoring Algorithm” ， 12/26/2005, http://webcourse.cs.technion.ac.il/236506/Winter2005-2006/ho/WCFiles/crypto-slides-22-factoring.4x2.pdf，p.625~627、p.677

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

[10] Lucas Algorithm ， http://primes.utm.edu/prove/prove3_2.html

[11] Miller-Rabin Algorithm， http://primes.utm.edu/prove/prove2_3.html

[12] Matthew E. Briggs，”An Introduction to the General Number Field Sieve”，April/17/1998，Blacksburg Virginia， http://scholar.lib.vt.edu/theses/available/etd-32298-93111/unrestricted/etd.pdf ,Ch5, p.51~70。

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

[14] Paul Zimmermann ， Inria Lorraine ， Nancy ， ”GMP-ECM:yet another implementation of the Elliptic Curve Method”，France。 http:/www.loria.fr/zimmerma/records/ecmnet.html

[15] Qi Cheng，”Primality Proving via One Round in ECPP and One Iteration in AKS” ， http://www.cs.ou.edu/~qcheng/paper/aksimp.pdf ，p.1~5

[16] Steven Byrnes，“The Number Field Sieve”， Math 129 ， May / 18 / 2005 。 http://modular.fas.harvard.edu/129-05/final_papers/Steve_Byrnes.pdf，p.3~9

[17] Seung Kook Park ， ”The Elliptic Curve Method” http://www.math.uiuc.edu/~duursma/Math595CR/ParS.pdf，p.2~5

[18] Wikipedia ， the free encyclopedia ， Trial Division、Pollard’s P-1、Williams P+1、ECM。

[19] Yves GALLOT，”Implementation of the AKS Algorithm” (Source Code) http://perso.wanadoo.fr/yves.gallot/src/