私 立 東 海 大 學
資 訊 工 程 與 科 學 系 碩 士 班
碩 士 論 文

指導教授：許玟斌 博士

找尋公開金鑰密碼系統RSA不動點之方法

The Method of Finding The Singular Points in RSA

研 究 生：張卜仁

中 華 民 國 九 十 三 年 七 月 十 日

摘要

網際網路的盛行，讓我們正式進入數位時代，資訊安全一直是人們討論的焦點，密碼學的蓬勃發展正代表對資料保護的重要性。隨非對稱密碼體制中的公開金鑰加密系統 RSA 演算法的提出(加密使用公開金鑰 PK,$RSA(X,PK,N)=X^{PK} mod\ N$，解密使用密鑰 SK, $RSA(X,SK,N)=X^{SK}\ mod\ N$)，形成數位時代一個算法公開、密鑰保密的公開加密體系。著眼將一個大數因數分解成兩個大質數的乘積非常困難，構造一個強度高的公開加密體系，讓資料在一定的時間和空間內受到安全的保護。RSA 自從提出以後已經廣泛使用在各行業中，包括攸關經濟的電子金融業、電子商務等。從文獻中已經證實，對於任一選擇的公開金鑰(PK)，至少有 9 個明文消息不會以加密來隱匿，即不動點（Singular Point）至少有 9 個。因此我們使用 RSA 演算法，選擇較小的質數進行加密的數值實驗，並觀察加密過程中產生的函數值後，再找大的質數來驗證後。發現了 RSA 的有效加解密空間只有$\lfloor\ (N/2)\ \rfloor$，我們利用二次同餘運算可以找出除了 0,1,$(N-1)$以外三種類型的不動點(6 個)，有相鄰的兩組正整數和一組二次同餘為 1 的不動點，其中還包括有一組加密過程中完全不產生變化的點，且只要找到其中一個不動點，透過多項式運算可以找出其他不動點。藉由我們的研究提供給 RSA 使用者避免重要資訊不小心曝露在不動點中，並尋求更好的解決方法。

關鍵詞：公開金鑰加密、不動點、資訊安全。

Abstract

There are many businesses using RSA algorithm [17] to protect important messages such as e-commerce, financial institution and many others. This paper discussed the singular points in RSA. B.Blakley, G.R.Blakley and I.Borosh [18] had shown that the number message of plaintext cannot be concealed in RSA to at least 9 cases. The number message X is a singular point if $X^{PK} \pmod N \equiv X$. Based on my experiment there are at least 9 singular points in RSA system for smaller numbers of p,q , PK and very big prime numbers. One interesting result I discovered is that the cipher text of RSA can be exist only as pairs. Hence, there are merely $\lfloor (N/2) \rfloor$ availability cipher spaces of RSA. From our research, We can conclude the followings (1) X is a singular point, if $X^2 \pmod N \equiv X$. (2) If X is a singular point, either $X-1$ or $X+1$ is also a singular point. (3)There are two singular points satisfying $X^2 \pmod N \equiv X^3 \pmod N \equiv X^4 \pmod N \equiv \ldots\ldots\ldots\ldots \equiv X^{PK} \pmod N \equiv X$. (4) We can find the other singular points from ($X^2 \pmod N$, $X^3 \pmod N$, $X^4 \pmod N$,……,$X^{PK} \pmod N$), if X is singular point.

Keywords: RSA, Singular Point, Security.

目 次

圖目次

表目次

第 1 章 導論

RSA 是目前廣泛使用在電子商務保存重要的解密金鑰的演算法，著眼兩個大指數質數的乘積容易，但是要分解卻有很大的難度下構造的公開金鑰加密體制。確實對於目前的資訊安全提供一定的保護和思考。隨著強大攻擊者不斷的提出破譯的演算法和硬體設備的演算效能提昇，RSA 也從原本單純的質數選擇，修正成大在 10^{75} 以上的強質數、公開金鑰(Public Key ,PK)的長度以及私密金鑰(Private Key , SK)的長度增加，造成使用者本身的運算開銷隨指數成長。雖然安全強度的增加到 1024 位元以後，目前在短時間內並沒有快速的破解方法。但是取模運算的同時，產生的週期性變化與加解密過程中不會因為金鑰而產生加解密變化的不動點($RSA(X,PK,N)=X^{PK} \pmod N \equiv X$)。劉尊全[18]就指出，B.Blakley、G.R.Blakley 和 I.Borosh 已經證實，對於任一選擇的密鑰，至少有 9 個明文消息不會以加密來隱匿，即不動點（Singular Point）至少有 9 個，如果 p、q 選擇不好可能會有 50%以上的消息不發生變化。

1.1 **RSA 概述**

RSA[21]是在 1977 年由 Ronald L. Rivest、Adi Shamir 與 Leonard M. Adleman 所發明的公開金鑰加密演算法。首先設定兩個大質數 p、q，然後兩數相乘 N=p*q，再算出ϕ (N)=(p-1)*(q-1)，並找到與ϕ (N)互質的公開金鑰(PK)，計算出私密金鑰(SK)，以進行加解密的運算。主要演算過程如下，加解密運算過程如圖 1。

加密函數： $RSA(X,PK,N) = X^{PK} \bmod N \equiv Y$

解密函數： $RSA(Y,SK,N) = Y^{SK} \bmod N \equiv X$

其中：

(1) *X* 是未加密前的明文（保密），*Y* 為密文。

(2) *N* 是模數。$\phi(N)=(p-1)*(q-1)$ 。

(3) $gcd(PK, \phi(N))=1$; $PK*SK \pmod{\phi(N)} \equiv 1$

(4) 公開金鑰與模數為（*PK,N*）是所有人皆可以取得的資訊；私密金鑰(*SK*)是保密的，只有使用者本身知道。

(5) 知道 *N,PK* 而不知道 *p,q* 很難算出 *SK*。

(6) *X* 和 *Y* 是在*[0~(N-1)]*之間的正整數。(如果明文是符號會先編譯成數字)

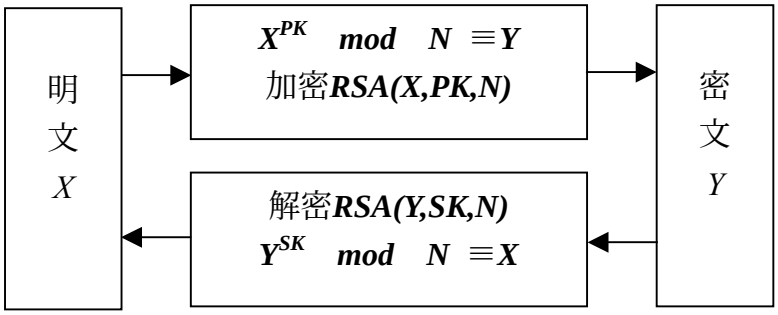

圖 1. RSA 演算法加解密過程

1.1.1 頻率

所有攻擊密碼系統的方法中，最常使用的就是頻率攻擊法，如無法有效的阻擋這類型的攻擊，將使整個加密系統產生不可預期的安全性問題。假設所有需要保護的明文資料都是英文字符，雖然經過加密後產生的密文資料中依舊會保有原來明文中的頻率特性，在 RSA 演算法中無法有效解決這樣的問題，為了更加了解這樣的內容，我們蒐集了 CNN 新聞網上發表的十篇文章內容加以分析統計，發現排除標點符號外，英文字母大小寫視為相同的狀況下仍舊是字母 E 出現的頻率最高，結果如表 1，將之繪製成統計圖後會發現一頻如圖 2。依照此頻率圖，如果蒐集夠多的密文資訊，統計出現頻率後，無須經過演算法的運算過程依然可以找出一定程度的明文資訊。

表 1.英文字母出現頻率統計表

字母	總和百分比	百分比 1	百分比 2	百分比 3	百分比 4	百分比 5	百分比 6	百分比 7	百分比 8	百分比 9	百分比 10
A	8.22	8.40	9.42	8.20	8.69	8.19	8.07	8.02	8.16	8.21	8.23
B	1.41	1.38	1.46	1.41	1.40	1.48	1.45	1.44	1.38	1.37	1.37
C	3.68	3.62	3.32	3.53	3.65	3.76	3.65	3.67	3.75	3.76	3.61
D	4.16	4.22	4.94	4.37	4.23	4.16	4.15	4.11	4.09	4.07	4.16
E	12.19	12.10	10.40	11.85	11.90	12.13	12.24	12.20	12.25	12.33	12.33
F	1.75	1.72	1.97	1.81	1.79	1.68	1.73	1.78	1.76	1.74	1.72
G	1.86	2.00	1.54	1.81	1.83	1.90	1.78	1.82	1.85	1.86	1.92
H	4.28	4.38	3.78	3.88	4.09	4.26	4.28	4.17	4.39	4.33	4.35
I	7.63	7.72	9.29	8.58	8.02	7.62	7.46	7.62	7.50	7.49	7.49
J	0.30	0.37	0.16	0.23	0.40	0.32	0.33	0.31	0.28	0.27	0.25
K	0.90	0.90	0.95	0.68	0.91	0.91	1.00	0.96	0.90	0.87	0.86
L	4.11	4.23	4.37	4.27	4.32	4.28	4.11	4.06	4.03	4.07	4.01
M	2.35	2.33	2.43	2.10	2.20	2.33	2.38	2.35	2.40	2.40	2.37
N	7.44	7.44	7.08	7.61	7.37	7.44	7.44	7.51	7.44	7.44	7.42
O	7.24	7.16	7.13	7.32	7.11	7.32	7.30	7.28	7.23	7.18	7.26
P	2.07	2.00	1.92	1.99	1.99	2.01	2.11	2.09	2.06	2.08	2.14
Q	0.15	0.16	0.54	0.26	0.19	0.15	0.14	0.15	0.14	0.14	0.13
R	6.68	6.71	7.35	6.59	6.69	6.65	6.62	6.67	6.62	6.69	6.74
S	7.74	7.58	6.89	7.73	8.02	7.60	7.79	7.75	7.77	7.77	7.75
T	8.53	8.38	8.45	8.88	8.32	8.55	8.47	8.61	8.59	8.53	8.48
U	2.60	2.53	2.38	2.61	2.51	2.66	2.65	2.64	2.62	2.59	2.57
V	1.06	1.11	1.05	1.09	1.10	1.04	1.04	1.05	1.03	1.05	1.11
W	1.76	1.85	1.43	1.45	1.55	1.79	1.83	1.85	1.77	1.77	1.76
X	0.19	0.13	0.05	0.16	0.14	0.15	0.20	0.21	0.22	0.22	0.21
Y	1.60	1.48	1.67	1.51	1.47	1.52	1.65	1.59	1.66	1.65	1.62
Z	0.11	0.10	0.03	0.08	0.08	0.10	0.12	0.11	0.10	0.09	0.15
SUM	100	100	100	100	100	100	100	100	100	100	100

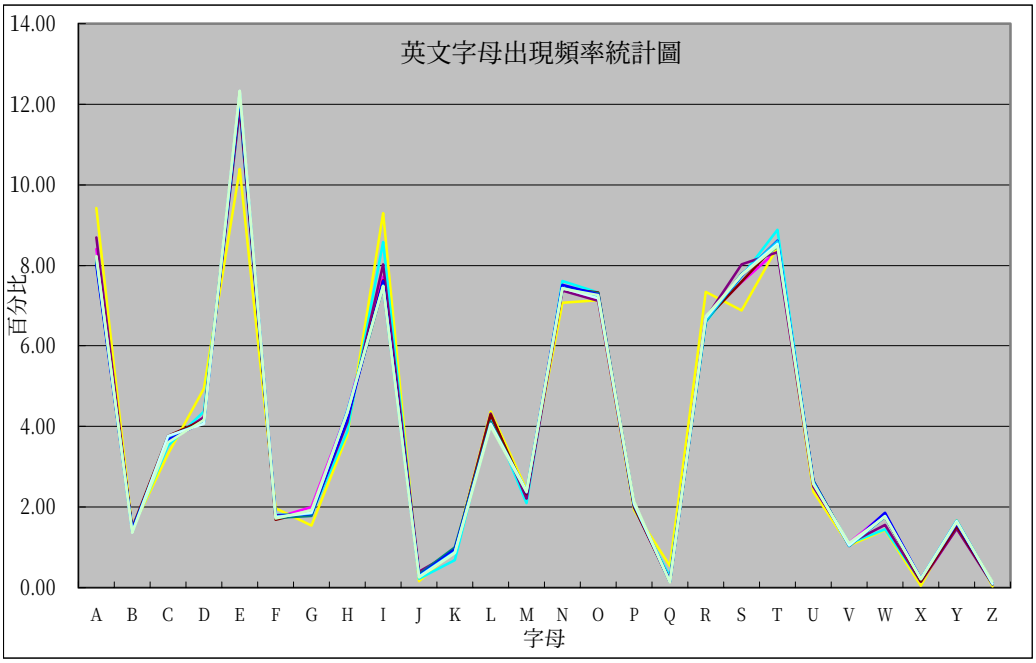

圖 2.英文字母出現頻率統計圖

1.1.2 週期性

週期性變化是同餘運算的一大特色，尤其 RSA 演算法中的最重要加解密觀念就是利用加密將週期打亂，然後利用解密將週期找回來的漂亮算法。當然，根據我們的研究中也發現了 RSA 的週期性變化，這裡介紹一個由東海大學李威和薛家羿得到的結論[19]，就是使用了週期性變化來解析 RSA 的典型方法，實作過程如下：

令 $N=p*q$

(1) 在明文空間中，任選一個明文當作明文 M。

(2) 在[0~$N-1$]中任選一個數當初始公鑰 PK_0。

(3) 利用 RSA 演算法進行加密 $M^{PK0} mod \quad N \equiv X_0$。

(4) 將 PK_0 每次加 1 成為 $PK_1, PK_2, \ldots\ldots, PK_n$，得到 $X_1, X_2, \ldots\ldots, X_n$。

(5) 直到 $X_n = X_0$。令 $PK_n - PK_0 = \phi(n)$。

(6) $\phi(n)$乘以整數倍數後會等於$\phi(N)$。

如果 p,q 選擇的不好，有可能在短時間內遭到此種週期性變化的攻擊，找出$\phi(N)$後就可以輕易算出 SK。但是如果可以將週期性變化的大小提升到一定的量，相信短時間內要使用這一種方法破譯 RSA 也是不可能的。

週期性其實是同餘運算的特性，經過同餘運算後所有元素會在一定的區間內，以 RSA 為例，一定會在[0~$N-1$]之間變化，不管原來的數字有幾位數。因此，如果使用 PK 不斷的對密文繼續加密，原理上面有可能回復到明文，只是運算量的大小而已。

1.2 研究想法

隨著安全性需求的增加，RSA 演算法不斷將 N 的長度從 512bits 增長到目前 1024bits，除了運算量隨指數增加之外，選擇強質數和加解密密鑰的限制、長度不斷

增加來抵抗種種的攻擊方式。但是，唯一無法改變的事實就是不動點，無論選擇強質數的 p 和 q 或者加解密密鑰長度的增加，因為取模運算的先天特性，其產生週期性變化是設計者當初始料未及的地方。

網際網路發達的數位時代，重要資訊的保存相當重要，目前所有演算法中，RSA 因為演算量的指數成長，被定位在保存少數的重要資料常用的方法，使用者一廂情願的認為這樣就安全了，事實上文獻中記載的 9 個不動點，代表了不管 p,q 的大小，選擇任一 $PK(RSA(X,PK,N)=X^{PK} \pmod N \equiv X)$，總有明文在經過加密後無法產生變化的數至少有 9 個，有可能更多的資訊暴露在攻擊者的面前而不自知。在目前尚未發現或並沒有比 RSA 更好的方法來保護重要資訊下，如果使用 RSA 演算法，我們要如何避免不動點呢？為什麼會有不動點的產生？是不是所有強質數構造 p,q 下都只有 9 個不動點？不動點和 p,q 的關係式？不動點之間又存在什麼樣的關係？會不會因為不動點的存在而影響 RSA 的安全強度？小質數產生的規律是不是可以推論到大質數？既然 RSA 演算法中一定會有不動點的出現，為什麼不再使用 RSA 的同時就公佈這一些不動點來避免重要資訊外洩呢？在在引起我們的研究興趣。

但是，大家都知道 RSA 最重要的安全性架構就是強的大質數，若要將所有大質數找出來就已經耗費大部分的時間，更遑論去觀察、歸納和解釋不動點的特性。基本上我們認為質數的特性在大小上面的表現應該是相同的。也就是說，如果在小質數的實驗中出現的規律，應該可以推論大質數也會出現相同的規律，除非我們找到的規律是特例。因此，我們尋找符合 RSA 演算法的小質數去做數值實驗，希望藉由小質數來找出大質數的不動點規律，更希望藉由小質數去發掘數論中難以讓人完全掌握的地方，是否可以同時解決質因數分解的歷史難題？

另外，如果目前並沒有更好的密碼系統保護重要的資料或者系統轉換的過度期間，仍然需要使用 RSA 演算法來保護資訊時，也希望能夠避免使用不動點進行加密。限於時間和設備的因素，僅就個人電腦設備可以實驗的範圍來設計，如果有大型電腦或者網格計算環境(Grid Computing Environment)的搭配，有機會挑戰 RSA 公佈的大數的質因數分解難題。

第 2 章 文獻探討

自從 RSA 提出以後，至今歷經二十多年，有無數的學者專家針對它提出多種的攻擊方法。到目前為止，並沒有一個有效的方法可以完全破譯 RSA，大多數的研究指出，如果在 p,q 選擇不當的時候，容易出現的週期性和封閉性問題，也都一一的有抵抗攻擊的方法，加大明文加密空間 N 和隨機選取大的強質數是目前仍然無法破解的原因之一，另一的原因就是無法證明破譯 RSA 比質因數分解的歷史 NP 問題還要簡單。我們就質數的定義和質因數分解的問題來了解 RSA 演算法的內涵，以方便了解這些 RSA 攻擊方法的主要精神。

2.1 質數的定義

對於任何一個大於 1 的正整數，如果除了 1 和本身之外，沒有其他的因數，則稱這個數為質數(Prime)，否則則稱為合數(Composite)。例如，2, 3, 5, 7, 11 等是質數，4, 6, 8, 9, 10 則是合數。著名的高斯「唯一分解定理」說，任何一個整數。可以寫成一串質數相乘的積。例如 28=7*22，31=1*31，84=22*3*7，10500=22 *3*52*7，10041=239*419，100149=3*7*19*251。

也就是說，任何數都可以表示成質數的乘積形式。下面的式子可以表達所有數和質數之間的關係：P 為所有質數的集合，任意正整數 a 皆可唯一以下列的式子來表示[9]：

$a = \prod_{P} p^{a_p}$ ， 而 p 是質數，對於某數 a 來說，大部分的指數項 a_p 的值皆為 0

2.2 質因數分解

兩個質數的乘積可以成為合數，在計算上相當容易，如果要將合數質因數分解就很難，歷經數百年也沒有人發現一個確定的演算法。隨著電腦運算能力的增強，也讓因數分解的範圍大大的提昇，目前已經有德國一個機構利用演算法順利分解 RSA 公司公佈兩年多的 RSA-576(十進位 174 位大的數字)。接下來介紹一些與因數分解有關的演算法[14]：

(1) 費馬演算法(Fermat's Algorithm)：假設 $n=p*q$，當 p,q 很接近$\sqrt{n}$時，假設 $p \geqq q>0$，則 $n=p*q=[[\frac{1}{2}*(p+q)]^2-[\frac{1}{2}*(p-q)]^2=t^2-s^2$，我們只要 t^2-n 是一個平方數 s^2，就可以解出 $p+q=t$ 且 $p-q=s$，就可以分解出 p,q。

例如：$\sqrt{8051}$ =89,90^2-8051=49=7^2,所以

(90+7)*(90-7)=8051 → 8051=97*83

(2) Kraitchik 演算法：假設 u^2-v^2 為 n 的倍數，則

$u^2 \equiv v^2 \pmod n \rightarrow n \mid (u^2-v^2)=(u+v)(u-v)$，假使 n 無法整除$(u+v)$或無法整除(u-v)的情況下，$gcd((u+v),n)$或 $gcd((u-v),n)$為 n 的一個真因數，如此即可質因數分解 n。

例 1：$\sqrt{13717421}$ =3703,$3705^2 \equiv 98^2 \pmod{13717421}$，

gcd(3705+98,13717421)=3803 → 13717421=3803*3607

例 2：$\sqrt{4633}$ =68,

$68^2=4624 \equiv -9 \pmod{4633}$,$69^2=4761 \equiv 128 \pmod{4633}$

$67^2=4489 \equiv -144 \pmod{4633}$,$7^2=4489 \equiv -144 \pmod{4633}$

$-9=(-1)3^2$,$128=2^7$,$-144=(-1)2^4 3^2$

$u^2 = 67^2 * 68^2 \equiv (-1)^2 * 2^4 * 3^4 = (2^2 * 3^2)^2 = v^2 \pmod{4633}$

$(67*68+36) * (67*68-36) \equiv 0 \pmod{4633}$

$(4592) * (4520) \equiv 0 \pmod{4633}$

gcd(4592,4633)=41, gcd(4520,4633)=113→4633=41*113

(3) 二次篩法(The Quadratic Sieve)：採用$f(x)=(x+[\sqrt{n}])^2-n$ 多項式為主，當 x 為整數時，有$f(x) \equiv (x+[\sqrt{n}])^2 \pmod{n}$，若我們找到一組 $x_1, x_2, x_3, \ldots\ldots, x_k$ 使得$f(x_1) * f(x_2) * \ldots\ldots * f(x_k)$為一個完全平方數$f(x_1) * f(x_2) * \ldots * f(x_k)=Y^2$，令 $X=f(x_1) * f(x_2) * \ldots\ldots * f(x_k) = (x_1+[\sqrt{n}]) * (x_2+[\sqrt{n}]) * \ldots\ldots * (x_k+[\sqrt{n}])$,則可得 $X^2 \equiv Y^2 \pmod{n}$，就可以去分解 n。其餘方法同 Kraitchik 演算法。

2.3 攻擊 RSA 的方法

雖然我們目前仍舊無法將 RSA 演算法的安全性證明等價於兩個大質數的分解上面，但是大多數的人認為縱使不等價也不會相差太多。既沒有證明也沒有反證的狀況下，RSA 自從提出之後，有許多學者針對它的特殊情況下蒐集密文和明文後，進行交叉比對後造成攻擊的效果，也有許多學者對於 p,q 之間的選擇不當大作文章，但是大多數的攻擊方法都有一些抵擋攻擊的措施，表 2 是由上航科技公司在網路上發表的文章[1]，主要介紹目前發表過的 RSA 攻擊方法。因為篇幅的關係，無法一一詳述其中的做法。

由表 2 的眾多學者攻擊 RSA 的方法中，我們知道 RSA 是由公開金鑰和密鑰構造一個非對稱密碼系統，接收者使用密鑰進行解密的暗門函數(Trap-door Function)，這樣的演算法遭受攻擊大約有下列幾種類型[17]：

(1) 質因數分解法：Pollard 等學者藉由 p,q 的選擇不當進行攻擊，因為$(p+1)$，$(p-1)$，

(q+1)或(q-1)其中有一個數字有很小的質因數時，容易遭受攻擊。一般學者會使用強質數來抵擋這樣的攻擊法。除此之外，尚有多重多項式二次篩選法(Multiple Polynomial Quadratic Sieve Factoring Method)和數域篩選法(Number Field Sieve Factoring Method)等方法進行 N 的質因數分解，根據 RSA 公司 2001 公佈的 RSA-576，直到 2003 年底德國一個機構才順利分解出，大約是十進位 174 個數字的組合，動用很多的人力和使用了許多電腦設備，這就是重要資料保護在可接受的時間和空間內的意義。

表 2.RSA 演算法攻擊方法

出處	攻擊主體	條件	結果	備註
Pollard	N	p-1 or q-1 has only small primes	get p or q	
Lehmanna	N	|p-q| is small	get p or q	
Williams	N	p+1 or q+1 has only small primes	get p or q	
Knuth	N	the ratio p/q is near a simple fraction	get p or q	
λ(p-1)	N	p-1=ap', q-1=aq' p'-1 or q'-1 has only small primes	get p or q	
Simmons-Norris	C	(1) p-1=ap', q-1=aq', p'-1 or q'-1 has only small primes (2)gcd(p-1,q-1)is largee (3)e has small order modulo(p-1)(q-1)	get M	
Hastad	C	e is small	get M	many parties
Wiener	e/N	e<N and d<4√N	get d and p and q	
Chen	e/N	e<N and d<4√N or (λ-d)< 4√N or |λ/2-d| < 4√N/2 or |mλ/h-d| < 4√N/h	get d and p and q	

N=p*q,e 是 PK,d 是 SK,M 是明文,C 是密文

資料來源[1]

(2) 選擇密文和明文攻擊法：由於同餘計算的定義，明文經過加密多次之後會在一

定的指數運算後可以恢復明文，這類型攻擊可以使用同一個數不斷的加大指數同餘計算，一直到恢復明文時，加密的指數就是與$\phi(N)$有關，東海大學李威和薛家羿曾經實作並得出結論 [19]。Simmons-Norris 等學者也提出，如$(p-1)$或$(p-1)$有很小的質因數時，可以蒐集一些密文並有效的恢復成明文。抵擋這類型的攻擊也是選擇強質數就可以。

(3) 小值密鑰和公開金鑰攻擊法：Wiener 等學者針對小值的密鑰，藉由連分法(Continued Fraction)，在 SK 小於$(1/3)*N^{0.25}$的情況下有效的解出 SK。最近，更有學者 Boneh 等將小值密鑰之範圍推至$N^{0.292}$。另外 Hastad 則針對數個相關的明文以同一公開金鑰加密，而其中 PK 又太小，當攻擊者截獲密文時，就有安全的顧慮。抵擋這類型攻擊的方法，就是加大 PK 和 SK 的大小或者將明文補白(Random Padding)的方式都會有效。

(4) 時間差攻擊法：RSA 在加解密的過程中，最大的運算量就是指數的同餘計算，因此藉由加解密的時間可以有效的猜測出 PK 和 SK。抵擋這類型的攻擊大多採用明文補白(Random Padding)，但相對的會造成整體運算時間的指數成長。

(5) 二次同餘法：假設$1<X<N$,且 $X^2 \mod N=1$ 時，可以知道X^2-1一定是 N 的倍數，那麼$(X-1)*(X+1)$也一定是 N 的倍數。根據質因數分解在整數範圍的唯一性可以知道$(X-1)$和$(X+1)$一定是 p 和 q 的倍數[21]，只要解出 X 也就等於 N 的質因數分解，我們也證明 X 就是一個不動點，相關證明請參考定理 7。

縱使如此，RSA 在經過這麼多的攻擊後，只要在 p,q 和 PK,SK 稍作適當的選擇，就可以使整個加密系統在合理的時間和空間內受到保護。根據文獻中顯示唯一不會有變化的就是不動點至少有 9 個，所以不動點的存在是 RSA 揮之不去的夢魘。

第 3 章 研究步驟及方法

在此，我們提出利用歸納法為主要研究步驟，先選擇較小的質數進行數值實驗，盡而力用實驗的結果找出規律後，再找更大的質數加以測試完成後進行數學證明。使用到的研究方法包括歸納法、蠻力法、同餘運算、歐幾里得(EUCLID)演算法及歐幾里得演算法延伸式，可以在知道 p,q 的情形下快速找到 9 個不動點。

3.1 研究步驟

侷限於本身電腦設備硬體等級(Intel Pentium II 233, 256MB RAM)和研究假設的想法，將研究分成下列步驟進行：

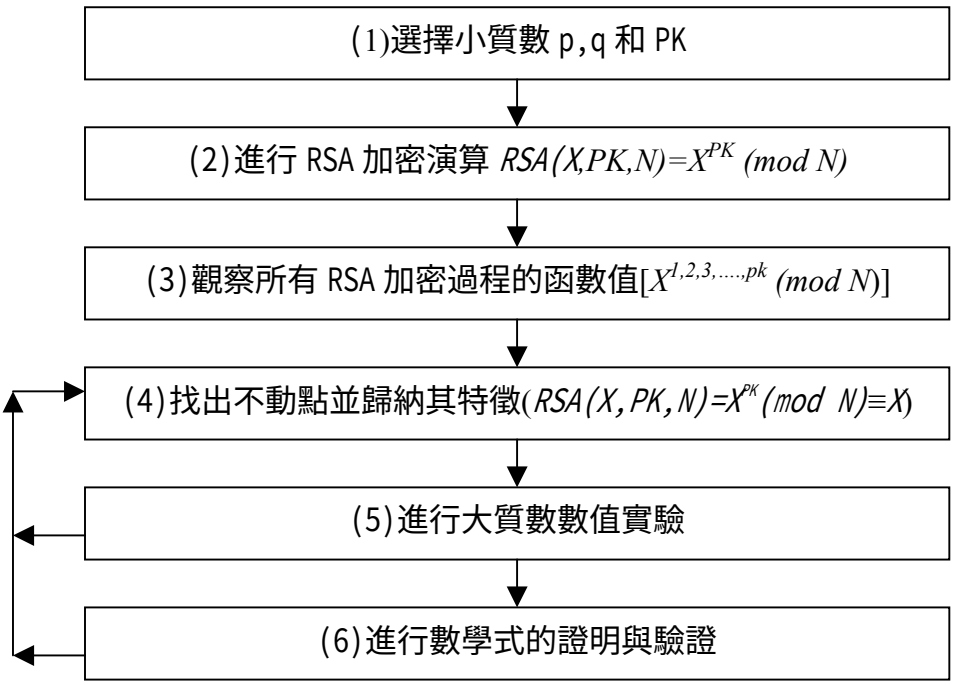

圖 3.研究步驟流程圖

(1) **選擇小質數 p,q 和 PK**：利用確定質數演算法找到小質數 p,q，並計算 $N=p*q$，

$\phi(N)=(p-1)*(q-1)$後，選擇小數的公開金鑰滿足 $gcd(PK, \phi(N))=1$。

(2) **進行 RSA 加密演算：**計算 $RSA(X,PK,N)=X^{PK} \pmod N, X=0,1,2,\ldots\ldots,(N-1)$

(3) **觀察 RSA 加密過程的函數值：**計算[$RSA(X,i,N)= X^i \pmod N, i=1,2,3\ldots,PK$]的值並記錄下來，觀察整體的變化過程並找出不動點 $X^{PK} \pmod N \equiv X$。

(4) **找出不動點並歸納其特徵：**分析不動點的特性，並歸納出其關係式。

(5) **進行大質數數值實驗：**找出比原來實驗還大的質數，代入歸納出來的關係式中進行數值實驗，檢驗是否符合關係式，如不符合則回到(4)繼續歸納。

(6) **進行數學式的證明與驗證：**如在大量數值實驗後符合的關係式，進行代數式的數學證明並找尋最佳解。如無法證明則回到(4)繼續歸納其他關係式。

3.2 研究方法

在 RSA 演算法中最常使用的就是同餘運算(Modular Arithmetic)，我們需要了解演算法的特性和規則，才有可能了解 RSA 的主要精神。另外，歐幾里得演算法主要是在於計算兩數的公因數，而使用延伸形式的歐幾里得演算法可以算出 $PK \bmod \phi(N)$ 的乘法反元素 SK，主要就是檢驗是否符合 $gcd(PK,\phi(N))=1$ 和找出 SK。

當然，RSA 主要的元素就是質數，而且是大質數，如何找到質數、大質數和強質數就是我們必須面對的主要難題。

3.2.1 同餘運算

同餘運算是 RSA 最主要的演算法則，也是目前密碼演算法常用的方法，最主要的原因在於大家認為同餘運算的結果會有非線性的表現，以及基本上的運算過程較為簡單，就此介紹一下同餘運算的一些特性[4]：

(1) 任意數 $a=\lfloor a/n \rfloor *n+(a \bmod n)$，其中$\lfloor a/n \rfloor$表示 a 除以 n 的整數商，$(a \bmod n)$則表示 a 除以 n 的餘數。

(2) $Z_n=\{0,1,2,\ldots\ldots,(n-1)\}$是小於 n 的正整數集合，對於每個 $x \in Z_n$，存在一個 z，使得 $x+z \equiv 0 \bmod n$，x,z 互為$(\bmod\ n)$加法反元素。

(3) 如果 $gcd(a,n)=1$，可以找到一個 $b \in Z_n$，使得 $a*b \equiv 1 \bmod n$，a,b 互為模 n 的乘法反元素。使用歐幾里得延伸形式演算法可以算出乘法反元素。

(4) $a \in Z_n$ 且 $b \in Z_n$，則有下列幾項運算規則

- $[(a \bmod n)+(b \bmod n)] \equiv (a+b) \bmod n$
- $[(a \bmod n)-(b \bmod n)] \equiv (a-b) \bmod n$
- $[(a \bmod n)*(b \bmod n)] \equiv (a*b) \bmod n$
- $[((k*a) \bmod n)+((k*b) \bmod n)] \equiv [k*(a+b)] \bmod n$

(5) Fermat 定理：若 p 為質數且 a 是無法讓 p 整除的正整數，則

$$a^{p-1} \equiv 1 \bmod p$$

(6) Euler 定理：$\phi(n)$是小於 n 但與 n 互質之正整數的數目，如 n 是質數則 $\phi(n)=n-1$，如 $n=p*q$ 且 p,q 都是質數則$\phi(n)=(p-1)*(q-1)$。定理說明如果 a 與 n 互質的話，則 $a^{\phi(n)} \equiv 1 \bmod n$。但是如 a 與 n 不互質，Euler 定理也成立，這也是 RSA 有效性的證明。

3.2.2 歐幾里得演算法

歐幾里得演算法[4]是數論中的一個基本技巧，這個演算法最主要的目的就是找出兩個正整數的最大公因數，也是 RSA 中檢驗 PK 和$\phi(N)$是否互質的主要工具。另外也可以使用延伸形式的歐幾里得演算法來求出一個數的乘法反元素(另一個數為模數)，也就是可以算出 SK。

歐幾里得演算法是以下面定理為基礎：對於任意非負整數 *a* 和任意正整數 b，*gcd(a,b)=gcd(b,a mod b)*，實際做法如下，結果如表 3：

EUCLID(a,b)

(1) *X←a ；Y←b*

(2) *if Y=0 return X=gcd(a,b)*

(3) *R=X mod Y*

(4) *X←Y*

(5) *Y←R*

(6) *goto (2)*

表 3.歐幾里得演算法實作

a mod b	X(a)	Y(b)	*b mod a*
	4199	5967	
4199	4199	1768	1768
663	663	442	442
221	221	0	0
a mod b	X(a)	Y(b)	*b mod a*
	1927	1951	
1927	1927	24	24
7	7	3	3
1	1	0	0

如果 $gcd(a,b)=1$，則 a 在模數 b 中存在一個乘法反元素。也就是說對正整數 $a<b$ 中存在一數 $a*a^{-1} \equiv 1 \bmod b$，如果 $b=\phi(N)$則 a 和 a^{-1} 就是 PK 和 SK，實際演算過程如下，結果如表 4：

延伸式 $EUCLID(a,b);a<b$

(1) $(X1,X2,X3) \leftarrow (1,0,b)$；$(Y1,Y2,Y3) \leftarrow (1,0,a)$

(2) *if* $Y3=0$ *return* $X3=gcd(a,b)$；*沒有反元素*

(3) *if* $Y3=1$ *return* $Y2= a^{-1} \bmod b$

(4) $Q=\lfloor X3/Y3 \rfloor$

(5) $(T1,T2,T3) \leftarrow (X1-Q*Y1,X2-Q*Y2,X3-Q*Y3)$

(6) $(X1,X2,X3) \leftarrow (Y1,Y2,Y3)$

(7) $(Y1,Y2,Y3) \leftarrow (T1,T2,T3)$

(8) *goto (2)*

表 4.歐幾里得延伸式演算法實作

Q	X1	X2	X3	Y1	Y2	Y3
-	1	0	185111	0	1	1951
94	0	1	1951	1	-94	1717
1	1	-94	1717	-1	95	234
7	-1	95	234	8	-759	79
2	8	-759	79	-17	1613	76
1	-17	1613	76	25	-2372	3
25	25	-2372	3	-642	60913	1
Q	X1	X2	X3	Y1	Y2	Y3
-	1	0	25199	0	1	247
102	0	1	247	1	-102	5
49	1	-102	5	-49	4999	2
2	-49	4999	2	99	-10100	1

3.2.3 如何找到質數

對於所有 RSA 的使用者而言，找到質數是一件非常重要的事情，因為 p,q 都是質數。雖然 PK 不是質數，但是因為 PK 和ϕ (N)必須互質，ϕ (N)是 2 和 4 倍數類型的合數，而大部分符合 PK 條件下的數都是質數，至於兩數互質的計算方面可以使用歐幾里得演算法即可。根據質數的定義，只要 n 無法被比 n 小的所有數整除就是質數，但是實作時並不需要測試所有小於 n 的數，如 n 是合數的話，必定有一個小於 $\sqrt{n}$ 的因數，且所有合數都至少有 2 個以上的質因數，所以使用小於 $\sqrt{n}$ 的所有奇數去測試即可。一般而言，我們會利用下列的步驟來找到確定的質數：

(1) 隨機挑選奇整數 n 當候選質數

(2) *for* $i=3$ *to* $\sqrt{n}$，i 是奇數

(3) 檢查 n 是否被 i 整除

(4) 如成立則 n 是合數，重新回到 (1)

(5) 都不成立 n 就是質數

這樣的方法算出來的 n 一定是質數，但在運算的量上面卻是不可行的，如 n 是 2^{100} 大的數字需要大約 2^{50} 的運算量才能確定是否為質數。形成在計算複雜度上面的困難，必須要有更有效的演算法來確認大質數，RSA 在軟體上面才有實現的可能。

3.2.4 如何找到大質數

大質數的尋找一直是數論中迷人的地方，目前為止並沒有一個固定的關係式可以算出隨機的大質數，美國田納西大學馬丁校區[30]上面所公布的質數就是經過嚴

格檢驗的，目前公佈的最大質數是 $2^{20996011}$-1，大約有 6320430 個十進位的數字，檢驗過程需要大量的運算設備、人力以及時間。所以，在實際應用上我們會利用以下的步驟來找尋我們所需要的隨機大質數：

(1) 根據所要求的位數，隨機挑選整數 *n* 當候選質數

(2) 對 *n* 作質數檢驗(Primality Test)

(3) 如果 *n* 被驗證不是質數，則回到步驟(1)；否則輸出 n

這裡最重要就是步驟(2)質數檢驗，檢驗大質數的運算量太大，所以在實際運算時主要都使用機率式檢驗法，在一個合理可接受的範圍下(t)去判定質數，如果選擇愈多的數(t 愈大)去測試，則 n 為質數的機率愈大，也是運算複雜度和可行性之間的平衡點，以下就介紹最常用的兩種質數檢驗方法[10]：

(1) Fermat 檢驗法，主要定理：若 *n* 為質數且 *a* 是無法讓 *n* 整除的正整數，則 $a^{n-1} \equiv 1 \bmod n$。因此，如果我們找到有一個小於 *n* 的數不滿足 Fermat 定理，就可以判斷 *n* 不是質數。實際做法如下：

for i = 1 to t

do

隨機挑選 *a*，*a* 介於 *2* 和 *n-2* 之間

計算 $r \equiv a^{n-1} \pmod n$

if $r \neq 1$ *then output* "*n* 是合數"

end do

output "*n* 是質數"

(2) Miller-Rabin 檢驗法，主要定理：若存在 $x^2 \equiv 1 \bmod n$ 的解不是 1 或-1，則 n 一定不是質數。所以在 Fermat 檢驗中隨時檢驗 $x^2 \equiv 1 \bmod n$ 的解，實際做法如下：

for i = 1 to t

do

隨機挑選 *a*，*a* 介於 2 和 *n-2* 之間

計算 $r \equiv a^{n-1} \pmod n$，過程中

if $\exists x^2 \equiv 1 \pmod n$ *but* $x \neq \pm 1 \pmod n$ *then output* "*n* 是合數"

if $r \neq 1$ *then output* "*n* 是合數"

end do

output "*n* 是質數"

3.2.5 如何找到強質數

強質數是學者們提出來抵抗 RSA 攻擊法的主張，那怎樣的質數才是強質數呢？基本上強質數 p 就是質數，且在 *p-1* 或 *p+1* 有一個大的質因數，如果符合下列三項規則的質數 *p* 就是強質數[12]。說明如下：

(1) *p,p1,p2* 都是質數

(2) *p-1* 或 *p+1* 有一個大的質因數 *p1*

(3) *p1-1* 或 *p1+1* 有一個大的質因數 *p2*

我們將以上的描述化成簡單的算式，因為根據質因數分解在整數範圍的唯一性，如果存在一個大的質因數，必定存在一個小的質因數，所有質數最小的就是 2，所以我們將強質數的定義如下：

(1) 等級 1 質數：*p1* 是質數

(2) 等級 2 質數：*p2=2*p1+1* 或 *p2=2*p1-1*，*p2* 是質數

(3) 等級 3 質數：*p3=2*p2+1* 或 *p3=2*p2-1*，*p3* 是質數

也可以將算式中的 2 換成小數 *a* 即：*p2=a*p1+1* 或 *p2=a*p1-1*，*p2* 是質數，以此類推 *p3,p4,…….*。其中 *a∈(2,4,6,……..)*。那等級 4,5,…….的強質數是否有更高的強度呢？答案是否定的。一則因為計算複雜度的增加造成計算量的指數成長無法在短時間內實現；二則強質數的數量會隨數字長度和等級的增加而減少，使得隨機選取的範圍縮小，反而容易造成 RSA 更不安全。所以，選擇強質數時一般建議在等級 3 左右即可。我們透過程式計算出強質數，結果如下：

RSA 強質數產生：*0<Level 1 質數 p1<1001*

(1) **Strong Prime Level 2**(*p+:p2=2*p1+1*):5,11,23,47,59,83,107,167, 179,227, 263,347,359,383,467,479,503,563,587,719,839,863,887,983,1019,1187,1283, 1307,1319,1367,1439,1487,1523,1619,1823,1907

(2) **Strong Prime Level 3**(*p++:p3=2*p2+1*):23,47,167,359,719,1439,2039,2879

(3) **Strong Prime Level 2**(*p-:p2=2*p1-1*):13,37,61,73,157,193,277,313,397, 421,457,541,613,661,673,733,757,877,997,1093,1153,1201,1213,1237,1321, 1381,1453,1621,1657,1753,1873,1933,1993

(4) **Strong Prime Level 3**(*p--:p3=2*p2-1*):73,313,1321,1753,1993,2473,3313

第 4 章 實驗結果與討論

4.1 實驗介紹

我們使用 Intel Pentium II 233, 256MB RAM 主記憶體, HP Vectra VE 電腦硬體進行數值實驗，軟體部分使用 SUN 公司出版之 JAVA 2 v1.4.2 和 Microsoft 出版之 Excel2000 及 JavaScript 2.0，所有測試的程式都是自行設計完成，使用的演算法包括蠻力法(Brute Force)、RSA 演算法、歐幾里得演算法等。

希望可以利用小質數的測試結果推導出大質數的規律。因此使用一般電腦硬體來進行數值實驗，實驗中所需的小質數部分採用確定法進行檢驗，大質數部分則利用 RSA 公司(http://www.rsasecurity.com/)公佈之結果，以及使美國田納西大學馬丁校區[30]上面所公布的質數代入 RSA 演算法中的 p,q，進行數值實驗。

4.2 實驗結果

我們在經過大量的小質數數值實驗後，希望藉由實驗的結果，找出 RSA 在加解密過程中的變化規律，並想要一窺不動點的真相。因此我們使用蠻力法進行實驗的數據結果來歸納小質數的規律，然後進行數學證明與大質數的驗證工作，發現三次同餘和二次同餘運算即可找到不動點。並且只要找到一個不動點後，其他的不動點也可以順利地找出來。

4.2.1 蠻力法尋找不動點

首先，我們選擇小質數 p,q 和 PK 進行 RSA 演算法的加密過程($Y \equiv X^{PK} \bmod N$)，利用蠻力演算法(Brute Force)算出所有加密後的值，將加密過程中產生的數值紀錄下來進行觀察並找出不動點。接著利用相同的 N(p,q 相同)和不同的 PK 進行實驗，試著找出共同的不動點。結果如表 5：

表 5.利用蠻力演算法求 RSA 加密過程

X	p	q	PK	N	X^1	X^2	X^3	X^4	X^5	Y
1	3	7	5	21	1	1	1	1	1	1
2	3	7	5	21	2	4	8	16	11	11
3	3	7	5	21	3	9	6	18	12	12
4	3	7	5	21	4	16	1	4	16	16
5	3	7	5	21	5	4	20	16	17	17
6	3	7	5	21	6	15	6	15	6	6
7	3	7	5	21	7	7	7	7	7	7
8	3	7	5	21	8	1	8	1	8	8
9	3	7	5	21	9	18	15	9	18	18
10	3	7	5	21	10	16	13	4	19	19
11	*3*	*7*	*5*	*21*	*11*	*16*	*8*	*4*	*2*	*2*
12	*3*	*7*	*5*	*21*	*12*	*18*	*6*	*9*	*3*	*3*
13	*3*	*7*	*5*	*21*	*13*	*1*	*13*	*1*	*13*	*13*
14	*3*	*7*	*5*	*21*	*14*	*7*	*14*	*7*	*14*	*14*
15	*3*	*7*	*5*	*21*	*15*	*15*	*15*	*15*	*15*	*15*
16	*3*	*7*	*5*	*21*	*16*	*4*	*1*	*16*	*4*	*4*
17	*3*	*7*	*5*	*21*	*17*	*16*	*20*	*4*	*5*	*5*
18	*3*	*7*	*5*	*21*	*18*	*9*	*15*	*18*	*9*	*9*
19	*3*	*7*	*5*	*21*	*19*	*4*	*13*	*16*	*10*	*10*
20	*3*	*7*	*5*	*21*	*20*	*1*	*20*	*1*	*20*	*20*

其中 $N=p*q, Y \equiv X^{PK} \bmod N,\ gcd(PK, \phi(N))=1$

經過多組 N 和 PK 的數值實驗後，我們發現明文 X 和經過二次同餘加密後的密文 Y($Y \equiv X^2 \bmod N$)之間成對的對稱存在，結果如圖 4 和圖 5。由圖中可以明顯的看出，雖然 N 的大小不一，但是明文和密文之間的變化卻呈現對稱現象，對稱軸在

$X=(N/2)$。換句話說，令 $X+X'=N$，則 $X^2 \bmod N \equiv Y \equiv Y' \equiv X'^2 \bmod N$。且不動點($Y \equiv X^{PK} \bmod N \equiv X$)會與 $Y=1, Y=X$ 和 $Y=N-X$ 三條線相交。這是 RSA 在加密過程中產生的週期性變化，相關的證明請參考定理 1。

例如表 5 中：

$X=2, X'=19,\ 2^2 \bmod 21 \equiv 4 \equiv Y \equiv Y' \equiv 4 \equiv 19^2 \bmod 21$

$X=7, X'=14,\ 7^2 \bmod 21 \equiv 7 \equiv Y \equiv Y' \equiv 7 \equiv 14^2 \bmod 21$

定理1. *假設 $1 \leq X < N, X+X'=N$, 則*

$RSA(X,2,N)=RSA(X',2,N), RSA(X,2n,N)=RSA(X',2n,N)$, n 是大於 1 的整數

證明：

$X+X'=N$, 則 $X'=N-X$

$RSA(X',2,N) = RSA(N-X,2,N) \equiv (N-X)^2 \bmod N \equiv (N^2-2*N*X+X^2) \bmod N$

$\equiv X^2 \bmod N = RSA(X,2,N)$ 得證#

$RSA(X,4,N)=X^4 \bmod N \equiv [(X^2 \bmod N) * (X^2 \bmod N)] \bmod N$

$\equiv [(X'^2 \bmod N) * (X'^2 \bmod N)] \bmod N = RSA(X',4,N)$

同理 $RSA(X,6,N) = RSA(X',6,N)$

……

$RSA(X,2n,N)=RSA(X',2n,N)$, n 是大於 1 的整數，得證#

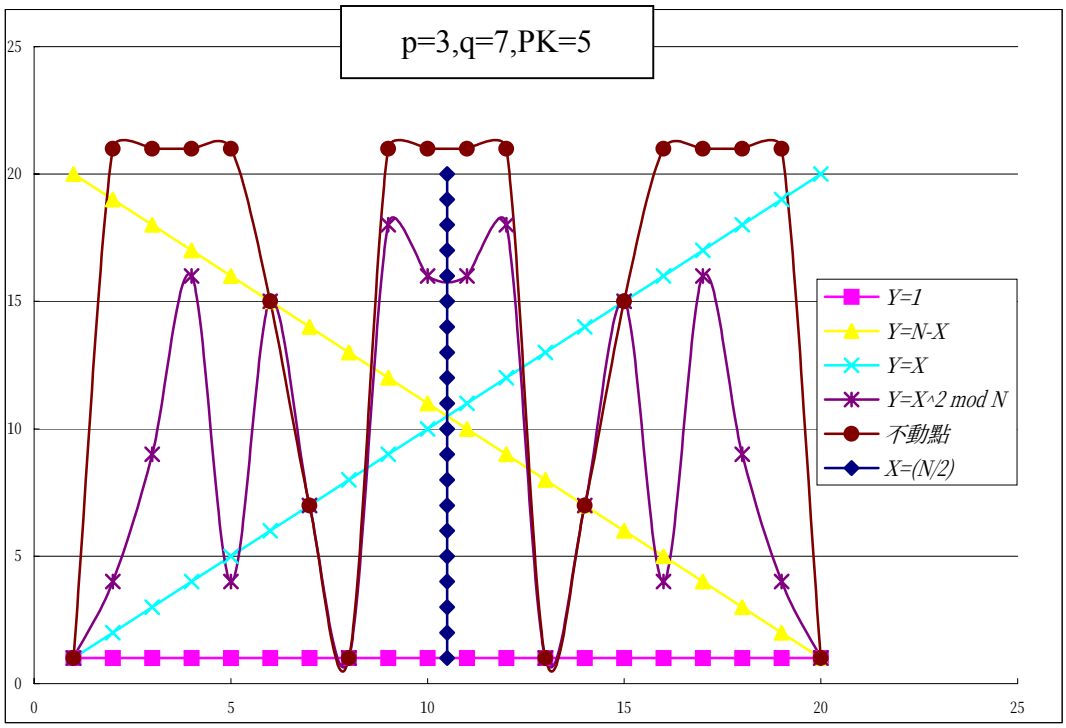

圖 4.RSA 演算法加解密過程對稱圖 1(二次同餘)

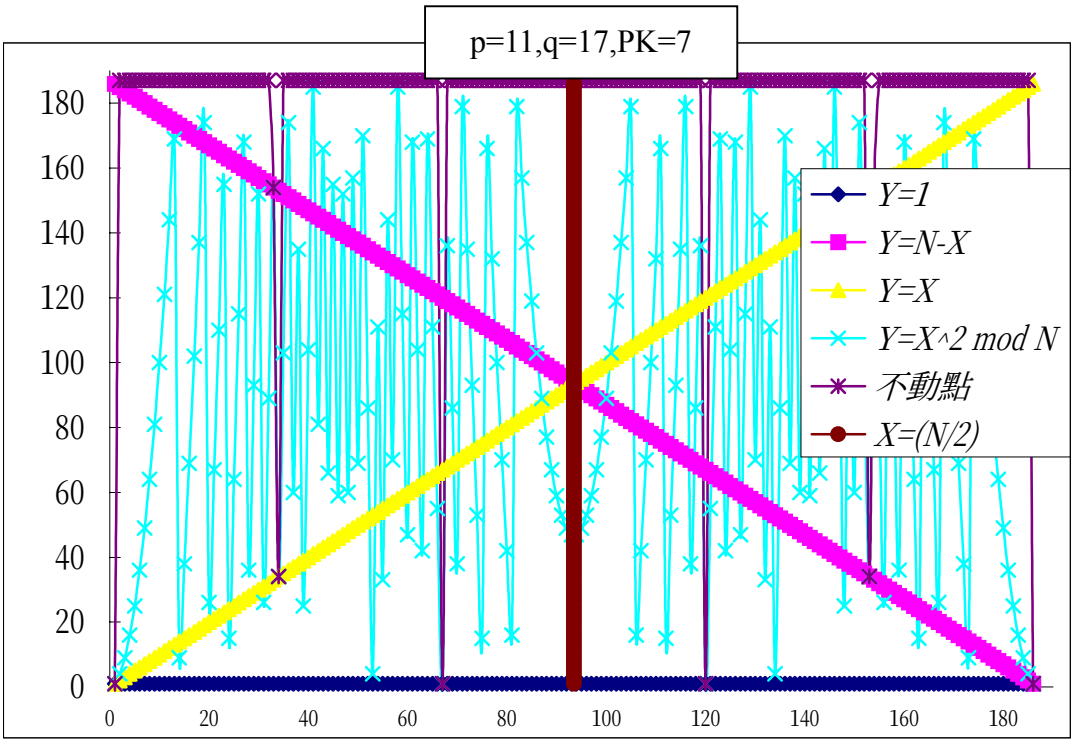

圖 5.RSA 演算法加解密過程對稱圖 2(二次同餘)

和為 N 的明文經過二次同餘加密後的密文對($Y \equiv X^2 \ mod \ N$)會有相同的值，那麼三次同餘呢？當初 RSA 被提出時因為硬體設備的運算能力不佳，造成 RSA 的加密速度太慢，所以有學者就建議使用低指數加密法，如果 3 與$\phi(N)$互質，就可以當成 PK 使用，直到有人提出將密文求立方根的攻擊法時才修正成大指數加密。當然，隨著硬體效能的提昇，已經很少人提出三次同餘的方法進行加密。我們卻觀察到可以當成 PK 的三次同餘運算後，只要明文兩數和為 N(X+X'=N)則加密後的密文和也是 N(Y+Y'=N)，結果可由表 5 中清楚看出來，在表 6 中將和為 N 的數字放在一起統計分析觀察。可以更清楚的看出兩者之間的對稱關係，對稱軸也是在(N/2)，如果將之會製成統計圖表會看到相鄰兩數(和為 N)在(N/2)兩側依序存在，且兩數和為 N，結果如圖 6，相關證明請參考定理 2。

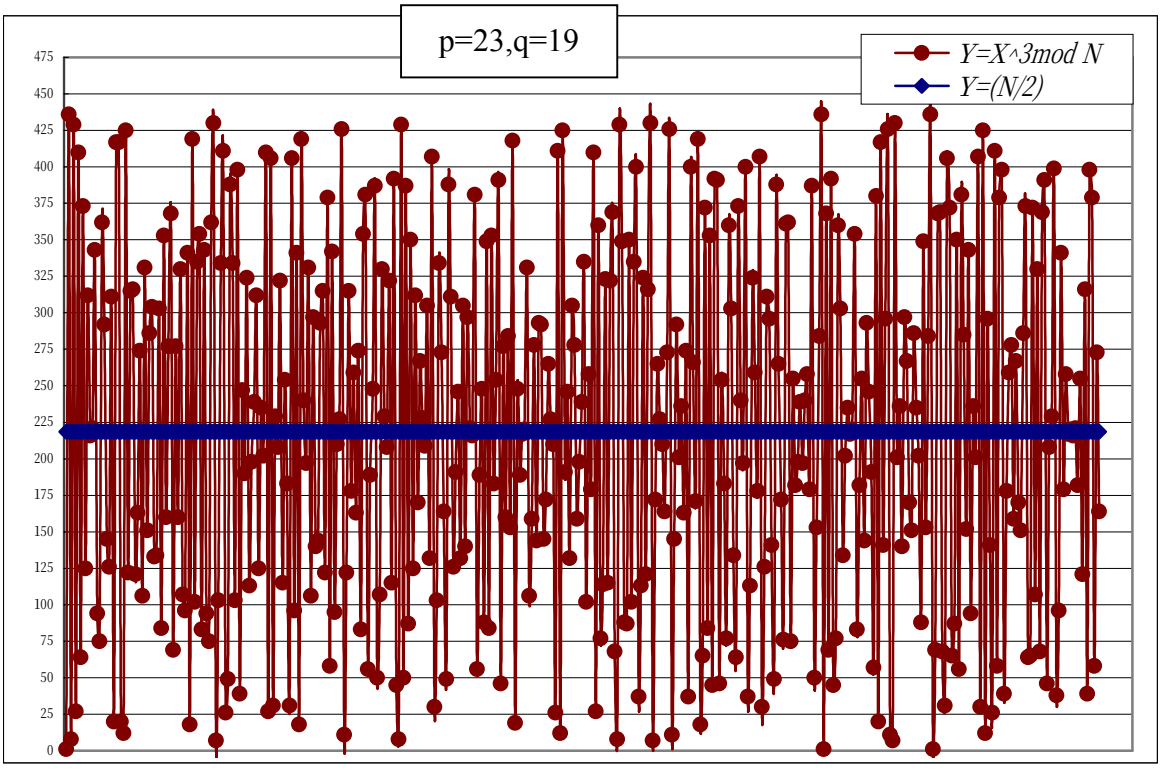

圖 6.RSA 演算法加解密過程對稱圖 3(三次同餘)

表 6.三次同餘運算 RSA 加密過程觀察(和為 N)

Y1	X	p	q	PK	N	X^1	X^2	X^3
218.5	1	23	19	29	437	1	1	1
218.5	*436*	*23*	*19*	*29*	*437*	*436*	*1*	*436*
218.5	2	23	19	29	437	2	4	8
218.5	*435*	*23*	*19*	*29*	*437*	*435*	*4*	*429*
218.5	3	23	19	29	437	3	9	27
218.5	*434*	*23*	*19*	*29*	*437*	*434*	*9*	*410*
218.5	4	23	19	29	437	4	16	64
218.5	*433*	*23*	*19*	*29*	*437*	*433*	*16*	*373*
218.5	5	23	19	29	437	5	25	125
218.5	*432*	*23*	*19*	*29*	*437*	*432*	*25*	*312*
218.5	208	23	19	29	437	208	1	208
218.5	*229*	*23*	*19*	*29*	*437*	*229*	*1*	*229*
218.5	209	23	19	29	437	209	418	399
218.5	*228*	*23*	*19*	*29*	*437*	*228*	*418*	*38*
218.5	210	23	19	29	437	210	400	96
218.5	*227*	*23*	*19*	*29*	*437*	*227*	*400*	*341*
218.5	211	23	19	29	437	211	384	179
218.5	*226*	*23*	*19*	*29*	*437*	*226*	*384*	*258*
218.5	212	23	19	29	437	212	370	217
218.5	*225*	*23*	*19*	*29*	*437*	*225*	*370*	*220*
218.5	213	23	19	29	437	213	358	216
218.5	*224*	*23*	*19*	*29*	*437*	*224*	*358*	*221*
218.5	214	23	19	29	437	214	348	182
218.5	*223*	*23*	*19*	*29*	*437*	*223*	*348*	*255*
218.5	215	23	19	29	437	215	340	121
218.5	*222*	*23*	*19*	*29*	*437*	*222*	*340*	*316*
218.5	216	23	19	29	437	216	334	39
218.5	*221*	*23*	*19*	*29*	*437*	*221*	*334*	*398*
218.5	217	23	19	29	437	217	330	379
218.5	*220*	*23*	*19*	*29*	*437*	*220*	*330*	*58*
218.5	218	23	19	29	437	218	328	273
218.5	*219*	*23*	*19*	*29*	*437*	*219*	*328*	*164*

其中 Y*1=(N/2)*，$X^2 \equiv X^2 \bmod N$，$X^3 \equiv X^3 \bmod N$，因篇幅無法列出 X 所有的數字

定理2. *假設 1≦X＜N,X+X'=N,則 RSA(X,3,N)+RSA(X',3,N)=N,*

RSA(X,2n±1,N)+RSA(X',2n±1,N)=N ,n 是大於 1 的整數

證明：

X+X'=N,則 X'=N-X

由定理 1 令 RSA(X,2,N)=RSA(X',2,N)=k

$RSA(X,3,N)= X^3 \bmod N \equiv [(X^2 \bmod N) * (X \bmod N)] \bmod N \equiv (k*X) \bmod N$

$RSA(X',3,N)= X'^3 \bmod N \equiv [(X'^2 \bmod N) * (X' \bmod N)] \bmod N \equiv (k*X') \bmod N$

$[RSA(X,3,N)+ RSA(X',3,N)] \bmod N \equiv [(k*X) \bmod N+(k*X') \bmod N] \bmod N$

$\equiv [k(X+X')] \bmod N \equiv 0$

∵1≦RSA(X,3,N)＜N 且 1≦RSA(X',3,N)＜N

∴RSA(X,3,N)+ RSA(X',3,N)=N 得證#

同理，由定理 1 令 RSA(X,4,N)=RSA(X',4,N)=k'

$RSA(X,5,N)= X^5 \bmod N \equiv [(X^4 \bmod N) * (X \bmod N)] \bmod N \equiv (k'*X) \bmod N$

$RSA(X',5,N)= X'^5 \bmod N \equiv [(X'^4 \bmod N) * (X' \bmod N)] \bmod N \equiv (k'*X') \bmod N$

$[RSA(X,5,N)+ RSA(X',5,N)] \bmod N \equiv [(k'*X) \bmod N+(k'*X') \bmod N] \bmod N$

$\equiv [k'(X+X')] \bmod N \equiv 0$

∵1≦RSA(X,5,N)＜N 且 1≦RSA(X',5,N)＜N

∴RSA(X,5,N)+ RSA(X',5,N)=N

同理 RSA(X,7,N)+ RSA(X',7,N)=N

……

RSA(X,2n±1,N)+RSA(X',2n±1,N)=N,n 是大於 1 的整數，得證#

4.2.2 不動點的特性

文獻中指出對於任一選擇的 PK，至少有 9 個明文消息不會以加密來隱匿。即不動點（Singular Point）至少有 9 個。經過小質數和蠻力法的尋找，我們可以順利找到許多不動點。我們也發現 N 相同但是 PK 不同的狀況下，會有一些明文始終都是不動點。如表 7 中 *N=19*23=437*，不同 PK 下仍有 1,114,115,208,229,322,323,436 等 8 個明文始終保持不變，如果加上 0，就完全符合文獻中所說不動點至少有 9 個的說法。

由表 7、表 8 中可以看到除了不動點和其他不動點的值之外，並沒有其他的數字出現在整個加密過程中。換句話說，就是不動點的加密變化過程中所出現的 RSA 函數值都是不動點。且循環週期很短，以表 7 為例，加密 7 次方後就恢復到明文了。這是一個觀察的結果，目前為止正努力尋求證明中。

我們蒐集了一些小質數產生的不動點進行數值實驗，將不動點和加密過程產生的函數值紀錄下來進行統計分析，結果如表 7 和表 8，可以看到幾個現象：

(1) 1 和 *N-1* 都是不動點，加上 0 共有三個所有 N 都會產生的不動點。

(2) 根據定理 2，不動點成對存在，且兩數的和為 N，如表 8 中的(6,29), (8,13), (208,229)。相關證明請參考定理 3。

(3) 這些不動點有一個很短的循環週期，只要三次方同餘運算後就恢復到明文。相關證明請參考定理 4。

(4) 這些不動點經過二次同餘計算後產生的函數值只有三種，就是 *1,X* 和 *N-X*。相關證明請參考定理 5。

(5) 這些不動點中都有相鄰的兩個數，且每一種 N 中都有兩組這樣的數字。相關證明請參考定理 6。如表 8 中的(14,15),(20,21),(6,7),(14,15),(114,323),(115,322)。

(6) 這些不動點中有一組數字 X 在加密過程中產生的值只有 X，並沒有產生其他

數值，也是就真正都沒有產生變化的完全不動點。如表 8 中的(15,21),(7,15) ,(115,323)等數。表 7 中的(115,323)

表 7.相同 N,不同 PK 的 RSA 不動點

X	p	q	PK	N	ϕ (N)	Y	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8	X^9	X^{10}	X^{11}
1	19	23	13	437	396	1	1	1	1	1	1	1	1	1	1	1	1
45	19	23	13	437	396	45	45	277	229	254	68	1	45	277	229	254	68
46	19	23	13	437	396	46	46	368	322	391	69	115	46	368	322	391	69
68	19	23	13	437	396	68	68	254	229	277	45	1	68	254	229	277	45
69	19	23	13	437	396	69	69	391	322	368	46	115	69	391	322	368	46
114	19	23	13	437	396	114	114	323	114	323	114	323	114	323	114	323	114
115	19	23	13	437	396	115	115	115	115	115	115	115	115	115	115	115	115
160	19	23	13	437	396	160	160	254	436	277	183	1	160	254	436	277	183
183	19	23	13	437	396	183	183	277	436	254	160	1	183	277	436	254	160
208	19	23	13	437	396	208	208	1	208	1	208	1	208	1	208	1	208
229	19	23	13	437	396	229	229	1	229	1	229	1	229	1	229	1	229
254	19	23	13	437	396	254	254	277	1	254	277	1	254	277	1	254	277
277	19	23	13	437	396	277	277	254	1	277	254	1	277	254	1	277	254
322	19	23	13	437	396	322	322	115	322	115	322	115	322	115	322	115	322
323	19	23	13	437	396	323	323	323	323	323	323	323	323	323	323	323	323
368	19	23	13	437	396	368	368	391	115	368	391	115	368	391	115	368	391
369	19	23	13	437	396	369	369	254	208	277	392	1	369	254	208	277	392
391	19	23	13	437	396	391	391	368	115	391	368	115	391	368	115	391	368
392	19	23	13	437	396	392	392	277	208	254	369	1	392	277	208	254	369
436	19	23	13	437	396	436	436	1	436	1	436	1	436	1	436	1	436
X	p	q	PK	N	ϕ (N)	Y	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8	X^9	X^{10}	X^{11}
1	19	23	17	437	396	1	1	1	1	1	1	1	1	1	1	1	1
114	19	23	17	437	396	114	114	323	114	323	114	323	114	323	114	323	114
115	19	23	17	437	396	115	115	115	115	115	115	115	115	115	115	115	115
208	19	23	17	437	396	208	208	1	208	1	208	1	208	1	208	1	208
229	19	23	17	437	396	229	229	1	229	1	229	1	229	1	229	1	229
322	19	23	17	437	396	322	322	115	322	115	322	115	322	115	322	115	322
323	19	23	17	437	396	323	323	323	323	323	323	323	323	323	323	323	323
436	19	23	17	437	396	436	436	1	436	1	436	1	436	1	436	1	436
X	p	q	PK	N	ϕ (N)	Y	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8	X^9	X^{10}	X^{11}
1	19	23	53	437	396	1	1	1	1	1	1	1	1	1	1	1	1
114	19	23	53	437	396	114	114	323	114	323	114	323	114	323	114	323	114
115	19	23	53	437	396	115	115	115	115	115	115	115	115	115	115	115	115
208	19	23	53	437	396	208	208	1	208	1	208	1	208	1	208	1	208
229	19	23	53	437	396	229	229	1	229	1	229	1	229	1	229	1	229
322	19	23	53	437	396	322	322	115	322	115	322	115	322	115	322	115	322
323	19	23	53	437	396	323	323	323	323	323	323	323	323	323	323	323	323
436	19	23	53	437	396	436	436	1	436	1	436	1	436	1	436	1	436

其中 $N=p*q, Y\equiv X^{PK} \bmod N$, $gcd(PK, \phi(N))=1$, $X^2\equiv X^2 \bmod N$……

表 8. RSA 演算法不動點加密過程

X	PK	N	ϕ (N)	Y	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8	X^9	X^{10}	X^{11}
1	11	35	24	1	1	1	1	1	1	1	1	1	1	1	1
6	11	35	24	6	6	1	6	1	6	1	6	1	6	1	6
14	11	35	24	14	14	21	14	21	14	21	14	21	14	21	14
15	11	35	24	15	15	15	15	15	15	15	15	15	15	15	15
20	11	35	24	20	20	15	20	15	20	15	20	15	20	15	20
21	11	35	24	21	21	21	21	21	21	21	21	21	21	21	21
29	11	35	24	29	29	1	29	1	29	1	29	1	29	1	29
34	11	35	24	34	34	1	34	1	34	1	34	1	34	1	34
X	PK	N	ϕ (N)	Y	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8	X^9	X^{10}	X^{11}
1	5	21	12	1	1	1	1	1	1	1	1	1	1	1	1
6	5	21	12	6	6	15	6	15	6	15	6	15	6	15	6
7	5	21	12	7	7	7	7	7	7	7	7	7	7	7	7
8	5	21	12	8	8	1	8	1	8	1	8	1	8	1	8
13	5	21	12	13	13	1	13	1	13	1	13	1	13	1	13
14	5	21	12	14	14	7	14	7	14	7	14	7	14	7	14
15	5	21	12	15	15	15	15	15	15	15	15	15	15	15	15
20	5	21	12	20	20	1	20	1	20	1	20	1	20	1	20
X	PK	N	ϕ (N)	Y	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8	X^9	X^{10}	X^{11}
1	53	437	396	1	1	1	1	1	1	1	1	1	1	1	1
114	53	437	396	114	114	323	114	323	114	323	114	323	114	323	114
115	53	437	396	115	115	115	115	115	115	115	115	115	115	115	115
208	53	437	396	208	208	1	208	1	208	1	208	1	208	1	208
229	53	437	396	229	229	1	229	1	229	1	229	1	229	1	229
322	53	437	396	322	322	115	322	115	322	115	322	115	322	115	322
323	53	437	396	323	323	323	323	323	323	323	323	323	323	323	323
436	53	437	396	436	436	1	436	1	436	1	436	1	436	1	436

其中 $N=p*q, Y\equiv X^{PK} \bmod N$, $gcd(PK, \phi(N))=1$, $X^2\equiv X^2 \bmod N$……

定理3. *假設 1<X<N 且 X+X'=N 且 RSA(X,PK,N)=X，則 RSA(X',PK,N)=X'*

證明：

X+X'=N,則 X'=(N-X),且 RSA(X,PK,N)=X

∵gcd(PK, ϕ (N))=1

∴PK 是奇數，可表示成 2n±1

由定理 2 得知

RSA(X,PK,N)+RSA(X',PK,N)=N

X+RSA(X',PK,N)=N,則 RSA(X',PK,N)=N-X=X' 得證#

定理4. *假設 1<X<N 且 RSA(X,3,N) =X，則 RSA(X,PK,N)=X，X 就是不動點*

證明：

$RSA(X,3,N)=X=X^3 \bmod N \equiv (X^2*X) \bmod N$

$RSA(X,5,N)=X^5 \bmod N \equiv [(X^2*X)*X^2] \bmod N \equiv (X*X^2) \bmod N \equiv X$

RSA(X,7,N)=X

……

RSA(X,PK,N)=X，∵PK 是奇數，得證#

定理5. *假設 1<X<N 且 RSA(X,2,N)=1 或 X 或 N-X ，則 RSA(X,PK,N)=X，X 就是不動點*

證明：

假設 1：RSA(X,2,N)=1

$RSA(X,2,N)=1 \equiv X^2 \bmod N$

$RSA(X,3,N)= X^3 \bmod N \equiv (X^2*X) \bmod N \equiv [(X^2 \bmod N)*(X \bmod N)] \bmod N$

$\equiv X \bmod N \equiv X$

同理 RSA(X,5,N)=X

…….

RSA(X,PK,N)=X 得證#

假設 2：RSA(X,2,N)=X

RSA(X,2,N)=X≡X^2 mod N

RSA(X,3,N)=(X*X^2)mod N≡[(X^2 mod N)*(X mod N)]mod N

≡(X*X)mod N≡X

同理 RSA(X,5,N)=X

…….

RSA(X,PK,N)=X 得證#

假設 3：RSA(X,2,N)=N-X

RSA(X,2,N)=N-X≡X^2 mod N

RSA(X,3,N)=(X*X^2)mod N≡[(X^2 mod N)*(X mod N)]mod N

≡[(N-X)*X]mod N≡(NX-X^2)mod N≡[(NX mod N)-(X^2mod N)]mod N

≡[0-(N-X)]mod N≡(X-N)mod N≡X

同理 RSA(X,5,N)=X

…….

RSA(X,PK,N)=X 得證#

∴只要二次同餘運算結果等於 1,X,N-X，X 就是不動點，表示如下：

如 X^2 mod N≡ $\begin{cases} 1 \\ X \\ N-X \end{cases}$ ，1<X<N 則 X 是 RSA 演算法中的不動點

定理6. *假設 1<X<N 且 RSA(X,2,N)=N-X ，則 X 和 X+1 都是不動點*

假設 1<X<N 且 RSA(X,2,N)=X ，則 X 和 X-1 都是不動點

證明：

假設 RSA(X,2,N)=N-X≡ X^2 mod N，根據定理 5 的假設 3，RSA(X,PK,N)=N-X

RSA(X+1,2,N)=$(X+1)^2$mod N≡(X^2+2X+1)mod N

≡[$(X^2$mod N)+(2X+1)mod N]mod N

≡(N-X+2X+1) mod N

≡(N+X+1) mod N

≡X+1

根據定理 5： RSA(X+1,PK,N)=X+1，X+1 是不動點

∴X 和 X+1 都是不動點，得證#

同理，假設 RSA(X,2,N)=X，RSA(X,PK,N)=X，X 是不動點

RSA(X-1,2,N)=$(X-1)^2$mod N≡$(X^2$-2X+1)mod N

≡[$(X^2$mod N)-(2X+1)mod N]mod N

≡(X-2X+1) mod N

≡[N-(X-1)] mod N

≡N-(X-1)

根據定理 5： RSA(X-1,PK,N)=N-(X-1)，X-1 是不動點

∴X 和 X-1 都是不動點，得證#

4.2.3 三次、二次同餘運算找不動點

根據以上觀察結果和證明，如果要找尋不動點，只要經過三次同餘運算和二次

同餘運算即可找到不動點，因為存在對稱關係，所以只要找到 3 個就會找到另外 3 個和為 N 的不動點。我們選擇稍大些的質數來作數值實驗。首先，我們使用三次同餘運算來找尋不動點。結果如表 9，可以清楚的看到第二個不動點的二次同餘沒有 1，只有在第一、三個不動點才會出現二次同餘為 1。而且三個不動點中有兩個字相鄰的數字，另外一個就是二次同餘為 1 的數字，相關證明請參考定理 4、定理 5 和定理 6。

接下來我們根據定理 5，利用二次同餘運算分別找出三種類型的不動點加以觀察，因為對稱性的關係，只要找到一個不動點即可找到另外一個對稱的點，結果如表 10、表 11 和表 12，由表中我們清楚的可以看到，如果 X 是符合二次同餘規律的不動點，就可以順利找出 X,X+1,X-1 與 N 的最大公因數就是 p,q 和 1，相關的的關係式可以表示如下，其中 a,b 都是正整數，相關證明請參考定理 7、定理 8。

$$X^2 \bmod N \equiv \begin{cases} 1 & \rightarrow X=a*p-1 \text{ 且 } X=b*q+1 \quad \rightarrow a*p-b*q=2 \\ X & \rightarrow X=a*p \text{ 且 } X=b*q+1 \quad \rightarrow a*p-b*q=1 \\ N-X & \rightarrow X=a*p \text{ 且 } X=b*q-1 \quad \rightarrow a*p-b*q=-1 \end{cases}$$

表 9. 利用三次同餘找出不動點($X^3 \bmod N \equiv X$)

p	q	N	X1	$X1^2$ %N	X2	$X2^2$ %N	X3	$X3^2$ %N
839	983	824737	131722	693015	131723	131723	263445	1
863	983	848329	183820	1	332254	516075	332255	332255
887	983	871921	227072	644849	227073	227073	417776	1
1019	1187	1209553	331174	1	439189	770364	439190	439190
1019	1283	1307377	247618	1	529879	777498	529880	529880
1019	1307	1331833	60121	1271712	60122	60122	120243	1
1019	1367	1392973	56046	1	668463	724510	668464	668464
1019	1487	1515253	200744	1	657254	857999	657255	657255
1283	1307	1676881	139848	1	768516	908365	768517	768517
1439	1367	1967113	27340	1939773	27341	27341	54681	1
1439	1487	2139793	44609	2095184	44610	44610	89219	1
1523	1487	2264701	629000	1	817850	1446851	817851	817851

其中 $X1^2 \%N \equiv X1^2 \bmod N$, $X2^2 \%N \equiv X2^2 \bmod N$, $X3^2 \%N \equiv X3^2 \bmod N$

表 10. 利用二次同餘找出不動點($X^2 \bmod N \equiv 1$)

p	q	N	X	(X,N)	(X+1,N)	(X-1,N)
1283	1307	1676881	139848	1	1307	1283
1019	1367	1392973	56046	1	1367	1019
1367	1439	1967113	54681	1	1439	1367
1523	1619	2465737	565032	1	1523	1619
1523	1823	2776429	240635	1	1823	1523
1523	1907	2904361	892477	1	1523	1907
1823	1907	3476461	413820	1	1823	1907
1619	1907	3087433	793311	1	1907	1619
1487	1907	2835709	499633	1	1907	1487
1999	1907	3812093	911545	1	1907	1999
1999	1997	3992003	1998	1	1999	1997
1999	1987	3972013	661670	1	1987	1999
1999	1979	3956021	395801	1	1999	1979
1999	1951	3900049	1137432	1	1951	1999
1999	1993	3984007	1327337	1	1993	1999

其中$(X,N)=gcd(X,N)$，$(X+1,N)=gcd(X+1,N)$ ，$(X-1,N)=gcd(X-1,N)$

表 11. 利用二次同餘找出不動點($X^2 \bmod N \equiv X$)

p	q	N	X	(X,N)	(X+1,N)	(X-1,N)
1283	1307	1676881	768517	1283	1	1307
1019	1367	1392973	668464	1019	1	1367
1367	1439	1967113	27341	1439	1	1367
1523	1619	2465737	950353	1619	1	1523
1523	1823	2776429	120318	1823	1	1523
1523	1907	2904361	446239	1523	1	1907
1619	1907	3087433	396656	1907	1	1619
1487	1907	2835709	249817	1907	1	1487
1999	1907	3812093	455773	1907	1	1999
1999	1979	3956021	197901	1999	1	1979
1999	1993	3984007	663669	1993	1	1999

其中$(X,N)=gcd(X,N)$，$(X+1,N)=gcd(X+1,N)$ ，$(X-1,N)=gcd(X-1,N)$

表 12. 利用二次同餘找出不動點($X^2 \bmod N \equiv N-X$)

p	q	N	X	(X,N)	(X+1,N)	(X-1,N)
1283	1307	1676881	768516	1307	1283	1
1019	1367	1392973	668463	1367	1019	1
1367	1439	1967113	27340	1367	1439	1
1523	1619	2465737	950352	1523	1619	1
1523	1823	2776429	120317	1523	1823	1
1523	1907	2904361	446238	1907	1523	1
1619	1907	3087433	396655	1619	1907	1
1487	1907	2835709	249816	1487	1907	1
1999	1907	3812093	455772	1999	1907	1
1999	1979	3956021	197900	1979	1999	1
1999	1993	3984007	663668	1999	1993	1

其中$(X,N)=gcd(X,N)$，$(X+1,N)=gcd(X+1,N)$ ，$(X-1,N)=gcd(X-1,N)$

定理7. *假設 $1<X<N$,且 $RSA(X,2,N)=1$ 或 X 或 $N-X$，則 X 或 $X+1$ 或 $X-1$ 的質因數中必有 p 或 q。*

證明：

假設 1：$RSA(X,2,N)=1=X^2 \bmod N$

$X^2-1=e*N$,其中 e 是正整數

$(X+1)*(X-1)= e*N$

∵$X<N$,且根據質因數分解在整數範圍的唯一性

∴$X+1=a*p$ 且 $X-1=b*q$，其中 a,b 是正整數，得證#

假設 2：$RSA(X,2,N)=X=X^2 \bmod N$

$X^2-X=e*N$,其中 e 是正整數

$X*(X-1)= e*N$

∵$X<N$,且根據質因數分解在整數範圍的唯一性

∴$X=a*p$ 且 $X-1=b*q$，其中 a,b 是正整數，得證#

假設 3：RSA(X,2,N)=N-X=X^2 mod N

*X^2+X=e*N,其中 e 是正整數*

X(X+1)= e*N*

∵X<N,且根據質因數分解在整數範圍的唯一性

*∴X=a*p 且 X+1=b*q，其中 a,b 是正整數，得證#*

定理8. *假設 1<X<N,且 RSA(X,2,N)=1 或 X 或 N-X，只要找到其中一個不動點，就可以找到另外兩種類型的不動點。*

證明：

*令 N=p1*q1=p2*q2=p3*q3*

假設 1：RSA(X,2,N)=X

根據定理 6，則 X-1 也是不動點且符合 RSA(X-1,2,N)=N-(X-1)

*根據定理 7，RSA(X,2,N)=X 則 X=a1*p1 且 X-1=b1*q1 則*

*X=a1*p1=b1*q1+1,則* ***a1*p1-b1*q1=1***

*根據定理 7，RSA(X',2,N)=1 則 X'+1=a2*p2, X'-1=b2*q2 則*

*X'=a2*p2-1= b2*q2+1,則 a2*p2-b2*q2=2*

2(a1*p1-b1*q1)=2= a2*p2-b2*q,令 p1=p2,則 a2=2*a1*

*X'=a2*p2-1=2*a1*p1-1=2*X-1,找到 X' 也是不動點且符合 RSA(X',2,N)=1*

假設 2：RSA(X,2,N)=N-X

根據定理 6, 則 X+1 也是不動點且符合 RSA(X+1,2,N)=X+1

然後再根據本定理假設 1 的做法找到 RSA(X',2,N)=1 的不動點

假設 3：RSA(X,2,N)=1

*根據定理 7，RSA(X,2,N)=1 則 X+1=a2*p2, X-1=b2*q2 則*

*X=a2*p2-1= b2*q2+1,則 a2*p2-b2*q2=2*

假設 X 是奇數，∵p2 奇數，∴a2 一定偶數則

*令(a2*p2-b2*q2)/2=2/2=1=a1*p1-b1*q1*

*令 p2=p1,則 2*a1=a2*

*X= a2*p2-1,則 X+1= a2*p2=2*a1*p1=2X'*

*X'=a1*p1=(X+1)/2,找到不動點符合 RSA(X',2,N)=X'*

接著可以找到 X'-1 也是不動點

*假設 X 是偶數，根據定理 2 可以找到 X''=N-X 也一定是不動點且 RSA(X'',2,N)=1，可表示成 a3*p3-b3*q3=2 且 X''是奇數，則可以利用前段假設*

*X''+1= a3*p3=2*a1*p1=2X'*

*X'=a1*p1=(X''+1)/2,找到不動點符合 RSA(X',2,N)=X'*

接著可以找到 X'-1 也是不動點

∴只要找到符合 RSA(X,2,N)=1 或 X 或 N-X 任一個不動點，即可找到其餘兩種類型的不動點，得證#

我們利用定理 7 和定理 8 可以歸納出二次同餘運算下的不動點關係式，且只要找到任一個不動點就可以找出另外兩種類型的不動點的證明。我們選擇找尋其中 $X^2 \bmod N \equiv X$ 的不動點，因為可以將關係式中的 *a*p-b*q=1* 視為 a 和 p 是模 q 的乘法反元素，利用歐幾里得延伸形式的演算法可以快速求解，即 $(a*p-b*q) \bmod q \equiv 1$ 則 $a*p \bmod q \equiv 1$，假設我們是 RSA 演算法的使用者，應該避免不動點的出現，就可以事先預防重要資訊的洩漏，結果如表 13，576Bits 的 N 為 RSA-576(RSA 公司公佈)。

另外，我們也不斷測試更大的質數來驗證關係式的正確性，結果是：

N 等於 241251bits 和 300315bits 時，已知 p,q 的狀況下，分別花費有關運算的時間：685656.0，847128.0 (1/1000 秒)、運算次數：1607 和 180(以歐幾里得延伸形式演算法計算)，幾乎可以說毫無困難就找到 6 個不動點。

表 13. 快速找到 RSA 不動點(已知 p,q)

N=166961,18bits,p=839,q=199 *不動點 1　X1=31043　X^2 mod N=135918* *不動點 2　X2=31044　X^2 mod N=31044* *不動點 3　X3=135917　X^2 mod N=31044* *不動點 4　X4=135918　X^2 mod N=135918* *不動點 5　X5=62087　X^2 mod N=1* *不動點 6　X6=104874　X^2 mod N=1* *時間:10.0 運算次數:36**	*N=563221,20bits,p=1811,q=311* *不動點 1　X=30788　X^2 mod N=1* *不動點 2　X2=532433　X^2 mod N=1* *不動點 3　X3=266217　X^2 mod N=266217* *不動點 4　X4=266216　X^2 mod N=297005* *不動點 5　X5=297005　X^2 mod N=297005* *不動點 6　X6=297004　X^2 mod N=266217* *時間:10.0 運算次數:16**
N=597629,20bits,p=1693,q=353 *不動點 1　X1=82956　X^2 mod N=1* *不動點 2　X2=514673　X^2 mod N=1* *不動點 3　X3=257337　X^2 mod N=257337* *不動點 4　X4=257336　X^2 mod N=340293* *不動點 5　X5=340293　X^2 mod N=340293* *不動點 6　X6=340292　X^2 mod N=257337* *時間:20.0 運算次數:48**	*N=38041,16bits,p=349,q=109* *不動點 1　X1=1745　X^2 mod N=1745* *不動點 2　X2=1744　X^2 mod N=36297* *不動點 3　X3=36297　X^2 mod N=36297* *不動點 4　X4=36296　X^2 mod N=1745* *不動點 5　X5=3489　X^2 mod N=1* *不動點 6　X6=34552　X^2 mod N=1* *時間:10.0 運算次數:4**

N=188198812920607963838697239461650439807163563379417382700763356422988859715234665485319060606504743045317
*388011303396716199692321205734031879550656996221305168759307650257059,**576Bits(RSA-576)***
p=472772146107435302536223071973048224632914695302097116459852171130520711256363590397527
q=398075086424064937397125500550386491199064362342526708406385189575946388957261768583317
不動點 1, X1 =1050000098667497889754148847861058771676624837784694739745385162748344181475717530780870994157485972873826632903780981862049411496523077260986743807039482495604638292451277 88
X1^2 mod N=1050000098667497889754148847861058771676624837784694739745385162748344181475717530780870994157485972873826632903780981862049411496523077260986743807039482495604638292451277 88

不動點 2, X2=1050000098667497889754148847861058771676624837784694739745385162748344181475717530780870994157485972873826632903780981862049411496523077260986743807039482495604638292451277 87
X2^2 mod N=8319880305385817486328235467554456263950107960094790872622484014815444156766291240723196119075614575793472472092529852999475117155342630578087627629227305560829547840512927 2

不動點 3,X3=8319880305385817486328235467554456263950107960094790872622484014815444156766291240723196119075614575793472472092529852999475117155342630578087627629227305560829547840512927 2
X3^2 mod N=8319880305385817486328235467554456263950107960094790872622484014815444156766291240723196119075614575793472472092529852999475117155342630578087627629227305560829547840512927 2

不動點 4, X4=8319880305385817486328235467554456263950107960094790872622484014815444156766291240723196119075614575793472472092529852999475117155342630578087627629227305560829547840512927 1
X4^2 mod N=1050000098667497889754148847861058771676624837784694739745385162748344181475717530780870994157485972873826632903780981862049411496523077260986743807039482495604638292451277 88

不動點 5 ,X5=1663976061077163497265647093510891252790021592018958174524496802963088831353258248144639223815122915158694494418505970599895023431068526115617525525845461112165909568102585 43
X5^2 mod N=1

不動點 6 ,X6=2180120681289161411213253011056131452816140417752156524831367612667997657990884067085513822499245152944793856945279965621018997809888142031779810441167519395216835083999851 6
X6^2 mod N=1
*時間:170.0 運算次數:157**

*：其中時間的單位為 1/1000 秒，運算次數以歐幾里得延伸形式演算法計算

第 5 章 結論與未來研究方向

5.1 結論

RSA 經過 PK 的指數取模運算後產生的密文變化，大家都認為一定是非線性的轉換，經過我們的證明得知並非完全如此。但我們並無法證明 RSA 的安全性和質因數分解是否等價，也無法證實 RSA 不安全。只是經過大量數值實驗後，我們歸納出下列幾項結論：

(1) *[0,N-1]* 所有之正整數經過 RSA 演算後兩兩成對存在，不動點也是成對存在，因此明文有效加密空間為⌊(*N/2*)⌋。

(2) 除了 *0,1,(N-1)*外，利用運算量較小的二次同餘及三次同餘運算($X^3 \bmod N=X$)即可找出下列 3 組(6 個)不動點，關係式如下。(*1<X<N*)

$$X^2 \bmod N \equiv \begin{cases} 1 & \rightarrow X=a*p-1 \text{ 且 } X=b*q+1 \quad \rightarrow a*p-b*q=2 \\ X & \rightarrow X=a*p \text{ 且 } X=b*q+1 \quad \rightarrow a*p-b*q=1 \\ N-X & \rightarrow X=a*p \text{ 且 } X=b*q-1 \quad \rightarrow a*p-b*q=-1 \end{cases}$$

其中 a,b 都是正整數

(3) 利用二次同餘運算找出一個大於 1 的不動點 X 後，經過多項式運算可以出其他 5 個。也就是說，只要我們找到一個不動點，其他不動點也呼之欲出了。

(4) 假設 *1<X<N*，則存在兩組(4 個)相鄰的正整數不動點*(X,X+1),(X,X-1)*。

$X^2 \bmod N \equiv (N-X)$且$(X+1)^2 \bmod N \equiv (X+1)$

$X^2 \bmod N \equiv X$且$(X-1)^2 \bmod N \equiv N-(X-1)$

(5) 假設 *1<X<N*，則存在一組(2 個)在加解密過程中完全不產生變化的完全不動點。$X^2 \bmod N \equiv X^3 \bmod N \equiv \cdots\cdots \equiv X^{PK-1} \bmod N \equiv X^{PK} \bmod N \equiv X$

(6) 已知 p,q 使用歐幾里得演算法可以快速找出 RSA 的 3 種類型(6 個)不動點。

(7) 除了本文定義的關係式中 9 個不動點之外，只要找出一個大於 1 的不動點 X 後，觀察 X 的加密過程產生的函數值，這些函數值也是不動點 ($X^2 \bmod N$, $X^3 \bmod N$, ……, $X^{PK-1} \bmod N$, $X^{PK} \bmod N$)。

5.2 未來研究方向

RSA 演算法從發表至今已經超過 25 年，一個演算法可以在這樣長的時間中面臨強大攻擊者的挑戰，仍然屹立不搖，其對於密碼學歷史的貢獻可見一斑，但是我們常常都只是去使用這樣人人都說好的演算法，卻忽略了其中隱藏的危機，隨著電腦硬體不斷的更新，我們需要使用長度更大的密鑰，卻也讓自己陷入指數增長的運算量中的 NP 問題。

數論的爭議、猜想與挑戰，是大自然界中人人都想了解的奧秘，面對數百年來的質因數分解和大質數尋找的的歷史難題下，建構 RSA 演算法的安全性，事實上並不恰當。經過證明，RSA 的有效加解密空間為目前的一半，如果在還沒有找到更好的演算法前，我們使用 RSA 時應該注意一下其先天的不足，並且正視週期性變化以及不動點的問題，盡量避免因此而產生的危機。

5.2.1 如何快速找到不動點(未知 p,q)

雖然我們已經找到不動點的規律，在知道 p 和 q 的狀況下，透過多項式的運算以及歐幾里德演算法延伸的運算，可以找到 $a \times p - b \times q = 1$ 的解，順利找到 6 個不動點。但是，p 和 q 在 RSA 演算法中是保密的，我們有沒有快速的數值運算方法找到不動點？。因此，如果可以找到一個演算法快速找出 RSA 的不動點，代表了 RSA 的 N 可以快速的被分解，如此的安全性就堪慮了。另外也可能解決數百年來質因數

分解的歷史難題。

5.2.2 如何找到超過 9 個以上的不動點

在相當多的文獻中，RSA 的不動點至少存在有 9 個。但是除了這 9 個以外呢？因為 p 和 q 是隨機選擇的質數，縱使是強質數也沒有數學證明只有 9 個。所以，是否存在第 10 個或第 11 個……。如果 p,q 選擇不當，不動點會有 20%或者更多的比率，目前我們僅就一定存在的 9 個不動點去定義，如果有更小的不動點存在，它是否也跟這 9 個不動點有相同的特性？可有快速的方法求出？

目前，我們蒐集到少數的不動點資料中顯示，只要找到一個不動點後，只要算出$(X^2 \bmod N, X^3 \bmod N, \ldots\ldots, X^{PK-1} \bmod N, X^{PK} \bmod N)$這一串的函數值，所有的值都是不動點，是否可以有數學的證明式？這是我們日後值得研究的課題之一。

參考文獻

中文部分

[1] 上航科技公司，2002， http://www.sunzone.com.tw/pw_system.htm。

[2] 于秀源，2001 年 9 月，”關於 RSA 加密方法不動點的注記”，杭州師範學院數學與應用研究所，計算機學報第 24 卷第 9 期，頁 998-1001。

[3] 于秀源，2002 年 5 月，”關於 RSA 加密方法不動點的注記(II)”，杭州師範學院數學與應用研究所，計算機學報第 25 卷第 5 期，頁 497-501。

[4] 巫坤品、曾志光譯，2001，密碼學與網路安全—原理與實務第二版，台北市：碁峰。

[5] 沈建男，民 90，JavaScript 到 JSP:Client/Server 端網頁程式設計，台北市：學貫。

[6] 阮韻芳編譯、Jonathan Knudsen 原著，1999，JAVA 密碼學，臺北市：歐萊禮。

[7] 吳逸賢、吳目誠編著，2001，精采 JavaScript 程式設計，北市：知城數位科技。

[8] 官大智，2002，”質數測試演算法簡介”，新訊傳播，第 8 期第 4 卷，頁 32-36。

[9] 胡鈞祥,陳榮傑，2001，”尋找大質數”，暨大電子期刊 http://emag.ncnu.edu.tw/，第 10 期。

[10] 唐文標，2000，”質數”，原載於數學傳播第四卷第四期 http://episte.math.ntu.edu.tw/articles/mm/mm_04_4_04/。

[11] 唐明月、陳英亮，民 73，機率學與應用，台北市：中興管理顧問公司。

[12] 唐世鋼、劉昭文，1994 年 12 月，”RSA 體制中高質量大素數的選擇”，電腦學刊，第 6 期，頁 5-8。

[13] 許介彥，民 91.11，”數不盡的質數”，科學教育(師大)254，頁 28-33。

[14] 陳正茂，民 87，” 密碼學與因數分解”，東海大學碩士論文。

[15] 張卜仁、許玟斌，民 92.12，” 公開加密演算法 RSA 週期性變化之研究”， 大葉大學資訊管理學系：第九屆資訊管理暨食物研討會，頁 226。

[16] 黃敏晃，2000，規律的尋求，臺北市：心理。

[17] 單懷靈，民 91.08，”RSA 密碼演算法攻擊法”，資訊安全論壇 7，頁 3-11

[18] 劉尊全，2001，數位時代的密碼技術與未來，台北市：松崗。

[19] 劉尊全、劉興剛等，民 92，破譯 RSA-Cracking RSA，台北市：全華。

[20] 賴松溪、韓亮、張真誠，1998，近代密碼學及其應用，台北市：松崗。

[21] 賴松溪、韓亮、張真誠，民 92，近代密碼學，台北市：旗標。

英文部分

[22] Boneh, D.; Durfee, G.; “Cryptanalysis of RSA with private key d less than $N^{0.292}$,“Information Theory, IEEE Transactions on , Volume: 46 Issue: 4 ,pp.1339 -1349,July.2000.

[23] Dan Boneh, Matthew Franklin,” Efficient Generation of Shared RSA Keys”, Journal of the ACM, Vol. 48, No. 4, July 2001, pp. 702–722.

[24] G.R. Blakley , I.Borosh,”Rivest-Shamir_Adlemam public-key cryptosystems do not always conceal messages”,Comp. And Math. With Appls. Vol.5. pp.169-178,1979.

[25] Hyun-Soo Hong,Ho-Kyu Lee,Hyang-Sook Lee,,”The better bound of private key in RSA with unblance primes”,Applied Mathematics Letters 139,pp351-362,2003.

[26] L.Hernamdez Encinas, J. Munoz Masque, A.Queiruga Dios,”Large Decryption Exponents in RSA”,Applied Mathematics Letters 16,pp293-295,2003.

[27] M. Wiener, “Cryptanalysis of short RSA secret exponents,” IEEE Trans. Inform. Theory, vol. 36, pp. 553-558, May 1990.

[28] R. L.Rivest, A.Shamir, L.Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[29] RSA Security Inc,2004, http://www.rsasecurity.com/.

[30] The University of Tennessee at Martin ,"The Prime Pages",http://primes.utm.edu/.

[31] Wiener M.J., "Cryptanalysis of short RSA secret exponents,"Information Theory, IEEE Transactions on , Volume: 36 Issue: 3 pp. 553 -558, May 1990 .

[32] William Stallings, Cryptography and Network Security : Principles and Practice Third Edition ,NJ : Prentice Hall,2003.

[33] Williams H.," A modification of the RSA public-key encryption procedure (Corresp.),"Information Theory, IEEE Transactions on , Volume: 26 Issue: 6, pp. 726 -729 , Nov 1980.