網路安全的理論與實務

楊中皇 著

第三章公開金鑰密碼系統

http://crypto.nknu.edu.tw/textbook/

第三章公開金鑰密碼系統

- RSA公開金鑰演算法
- RSA相關數論理論
- Diffie-Hellman公開金鑰演算法
- ElGamal公開金鑰加解密演算法
- 數位信封
- 中國剩餘定理
- 印度剩餘定理

RSA公開金鑰密碼系統

- 1977年美國麻省理工學院R. Rivest、A. Shamir、A. Adleman三位教授，共同發明
- 美國的專利是於1983年9月通過，專利號碼是4,405,829(剛好是個質數)，已於2000年到期
- 目前幾乎所有的瀏覽器皆內建此密碼系統
- 每個使用者皆有所謂公開金鑰(public key)及與之對應的私密金鑰(private key)，公開金鑰是兩個正整數(*E*、*N*)，私密金鑰則是一個數字(*D*)
- 加密公式 $C = M^E \bmod N$
- 解密公式 $M = C^D \bmod N$

RSA公開金鑰密碼系統

- 適用於公開網路，提供「數位信封」、「數位簽章」等
- 原理：兩個大數相乘容易，分解難
- 作法：1. 自己挑兩個大質數(例如150位數), P和Q
  2. 選個奇數E滿足 最大公因數GCD(E, (P-1) * (Q-1))=1

  公開金鑰(public key): 到處公開 E與N (= P * Q)

  3. 利用輾轉相除法,計算出D滿足 $D \times E = 1 \bmod (P-1) \times (Q-1)$ 餘數

  私密金鑰(private key): 自己保管 D

- 加密(encryption): 發文單位使用收文單位的公開資訊

  密文 $C = M^E \bmod N$

- 解密(decryption): 收文單位使用自己的秘密資訊

  明文 $M = C^D \bmod N$

RSA範例

- P = 7
- Q = 11 $\longrightarrow$ N = P * Q = 77
- E = 17 D = 53 (17* 53 = 901=1+15* (7-1)*(11-1))

- 公開金鑰: (E=17, N=77)
- 私密金鑰: (D=53)

- 加密: M = 2 $\longrightarrow$ $C = M^E = 2^{17} = 131072 = 18 \mod 77$
- 解密: C = 18 $\longrightarrow$ $M = C^D = 18^{53} = 2 \mod 77$

18^{53} = 338409674705212417609691900649852521476345945170 85140904192958791 68

大數分解有獎徵答

- 單筆最高可得美金貳十萬元整
- RSA-576 $10,000 (2003年12月分解)
- RSA-640 $20,000 (2005年11月分解)
- RSA-704 $30,000
- RSA-768 $50,000
- RSA-896 $75,000
- RSA-1024 $100,000
- RSA-1536 $150,000
- RSA-2048 $200,000

RSA-704 (212 digits):
74037563479561712828046796097429573142593188889231
28908493623263897276503402826627689199641962511784
39958943305021275853701189680982867331732731089309
00552505116877063299072396380786710086096962537934
650563796359

大數分解有獎徵答(續)

- Prize: US$200,000
- Decimal Digits: 617
- 25195908475657893494027183240048398571429282126204
  03202777713783604366202070759555626401852588078440
  69182906412495150821892985591491761845028084891200
  72844992687392807287776735971418347270261896375014
  97182469116507761337985909570009733045974880842840
  17974291006424586918171951187461215151726546322822
  16869987549182422433637259085141865462043576798423
  38718477447920739934236584823824281198163815010674
  81045166037730605620161967625613384414360383390441
  49526344321901146575444541784240209246165157233507
  78707749817125772467962926386356373289912154831438
  16789988504044536402352738195137863656439121201039
  71228221207203 57
- 大數分解有獎徵答

質數判斷 (Primality Testing)

- 質數： 一個正整數而只能被1及本身所整除
  2, 3, 5, 7, 11, 13, 17, ...

- 目前已知最大之質數： (2006年9月發現)

  $2^{32,582,657} - 1$ (9,808,358 位十進數字)

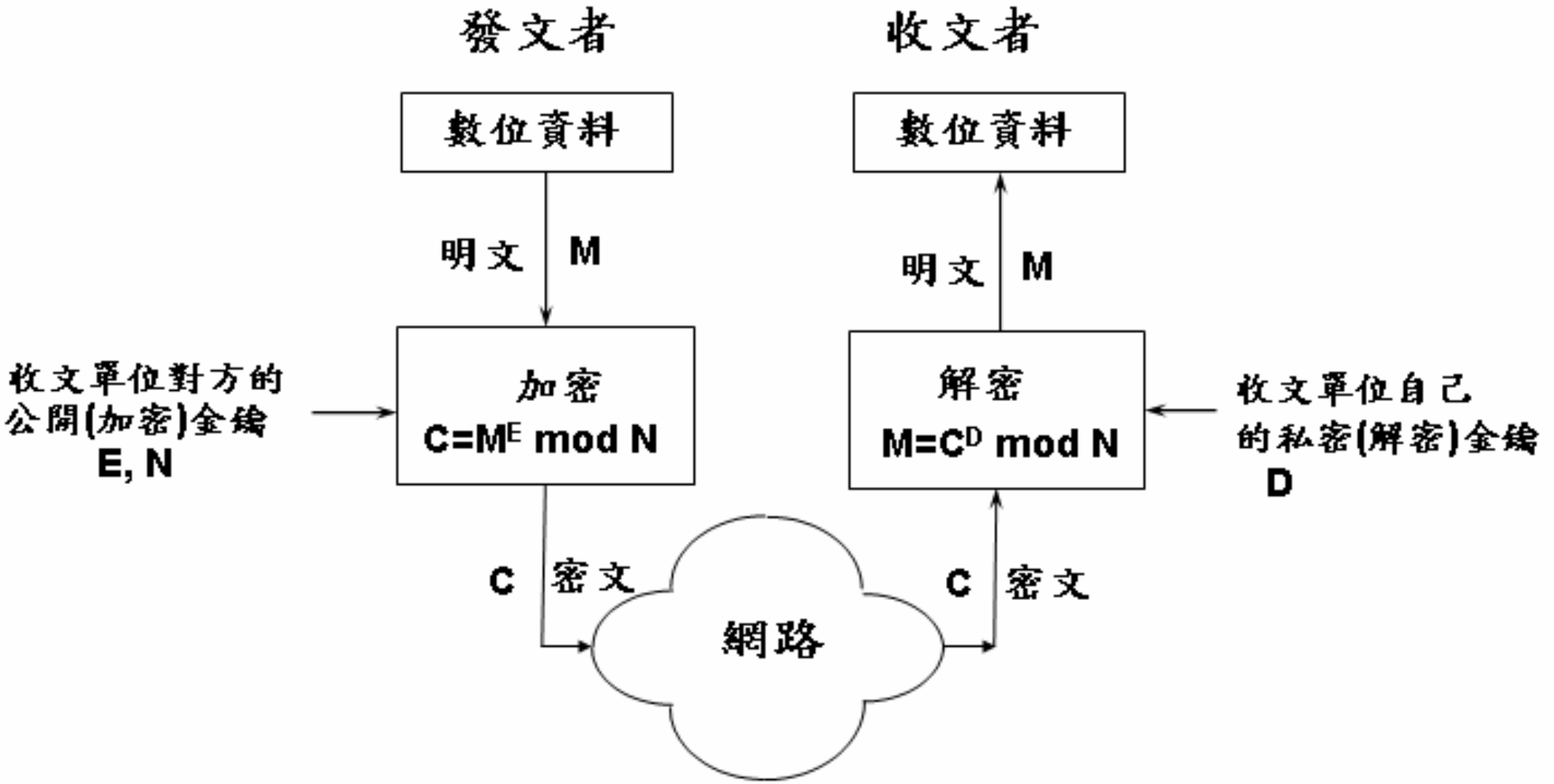
- 簡單的非質數判斷: 任何質數p一定滿足

  $a^{p-1} = 1 \bmod p$ (a 為小於p的質數)

例如 $p = 5,\ 2^{5-1}=2^4=16=1 \bmod 5$

$p = 9,\ 2^{9-1}=2^8=256=4 \bmod 9$

利用RSA密碼系統傳遞密文



快速計算A^B mod C平方再乘演算法

假設指數B可用k位元的二進制數字來代表，即 $B = (b_{k-1} b_{k-2} \ldots b_2 b_1 b_0)$

Step 1　$S \leftarrow 1$

Step 2　for　$i \leftarrow k\text{-}1$ downto 0

do　$S \leftarrow S \times S \bmod C$

If　$b_i = 1$

then　$S \leftarrow A \times S \bmod C$

範例計算$5^{10} \bmod 11$

- $5^{10} \bmod 11 = 9765625 \bmod 11 = 1$
- 指數$B = 10 = (b_3\, b_2\, b_1\, b_0) = (1010)$

平方再乘：

i	3	2	1	0
b_i	1	0	1	0
S	5	$5^2 \bmod 11 = 3$	$(3^2 \bmod 11) \times 5 \bmod 11 = 9 \times 5 \bmod 11 = 1$	1

計算最大公因數GCD: 輾轉相除法

- 希臘數學家歐基里得演算法(輾轉相除法)

- GCD (A, B) = GCD (B, A mod B)

Step 1. If $B = 0$, then return A

Step 2. $R \leftarrow A \bmod B$

$A \leftarrow B$

$B \leftarrow R$

Step 3. Goto Step 1

範例:

GCD (552,234) = GCD (234,84) = GCD (84,66) = GCD (66,18) = GCD (18,12) = GCD (12,6) = GCD (6,0) = 6

擴充歐基里得演算法

已知: T, N 求: X 滿足 $X \times T \mod N = 1$

X 又稱為 T 模 N 的乘法反元素(multiplicative inverse)，$X = T^{-1} \mod N$

Step 1. $A \leftarrow N$
$B \leftarrow T$
$X' \leftarrow 0$
$X \leftarrow 1$

Step 2. Q =floor($A \div B$) 註: ($A \div B$)的商
$R = A \mod B$ 註: ($A \div B$)的餘數

Step 3. If $R \neq 0$, then goto Step 4
If $X < 0$, then return $X + N$ else return X

Step 4. $A \leftarrow B$
$B \leftarrow R$
$X' \leftarrow X$
$X \leftarrow X' - Q \times X$
Goto Step 2

範例: 計算D滿足$D \times 5 \bmod 96 = 1$

A	B	X'	X	Q	R
96	5	0	1	19	1
5	1	1	-19	5	0

- 擴充歐基里得演算法第二次執行步驟3時，因$X = -19 < 0$，所以$-19 + N = 96-19 = 77$即為D的解

RSA的破解方式

一、暴力攻擊(brute force attack)：

- 逐一尋找可能的私密金鑰P、Q。
- 例如N是1024位元，可逐一嘗試512位元的所有質數，然後用N去除，看是否能除盡，若能整除的話那就是P，相對的也就能找出Q，這就是對RSA的暴力攻擊法。

二、數學上的攻擊(mathematical attacks)

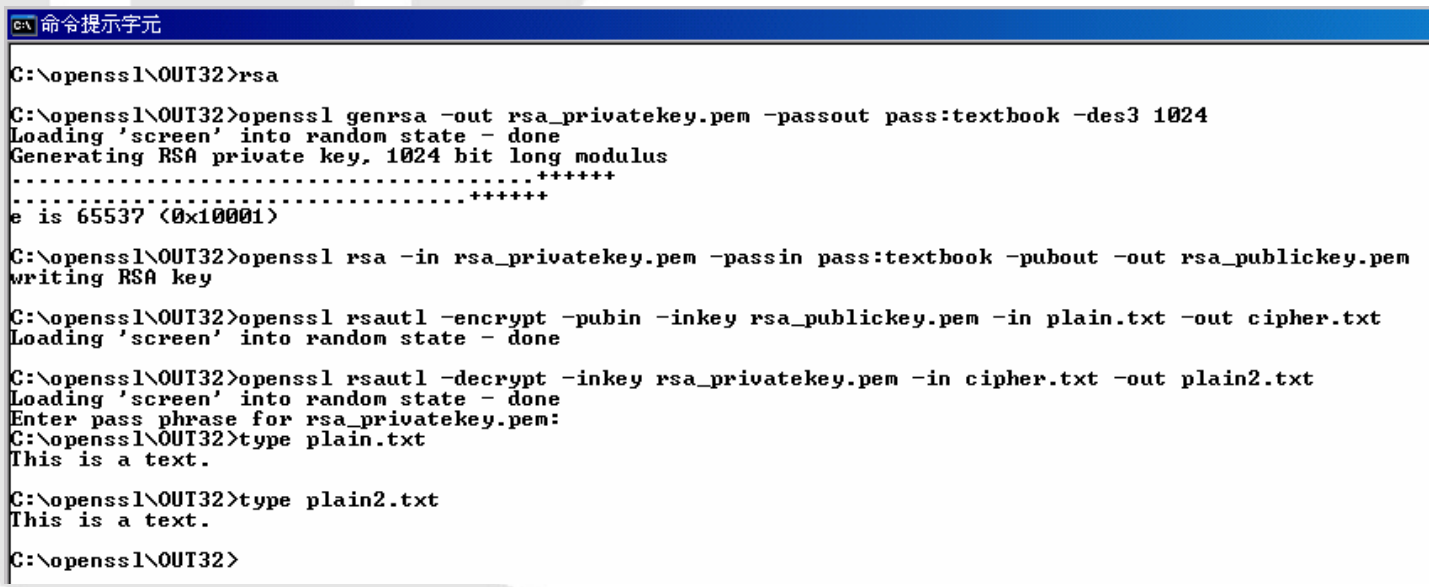
- 這種攻擊方式有很多種，例如尋找大數分解演算法快速計算出N，或者當私密金鑰$D < N^{0.292}$時，則研究人員可利用其他方式破解RSA

三、計時攻擊(timing attacks)

- 利用解密時間差異來猜測私密金鑰，從平方再乘演算法可看出指數(私密金鑰D) 的二進制數字中，若1很多則計算時間相對變慢，若1很少則計算時間相對會加快。因此計時攻擊就是利用解密時間的差異性來攻擊。

OpenSSL進行RSA加解密

- openssl genrsa -out rsa_privatekey.pem -passout pass:textbook -des3 1024
- openssl rsa -in rsa_privatekey.pem -passin pass:textbook -pubout -out rsa_publickey.pem
- openssl rsautl -encrypt -pubin -inkey rsa_publickey.pem -in plain.txt -out cipher.txt



OpenSSL測試RSA加解密效率

- openssl speed rsa1024 rsa2048 rsa4096

命令提示字元

```
C:\openssl\OUT32>rsaspeed

C:\openssl\OUT32>openssl speed rsa1024 rsa2048 rsa4096
To get the most accurate results, try to run this
program when this computer is idle.
First we calculate the approximate speed ...
Doing 655 1024 bit private rsa's: 655 1024 bit private RSA's in 12.63s
Doing 6553 1024 bit public rsa's: 6553 1024 bit public RSA's in 6.35s
Doing 81 2048 bit private rsa's: 81 2048 bit private RSA's in 9.98s
Doing 1638 2048 bit public rsa's: 1638 2048 bit public RSA's in 5.62s
Doing 10 4096 bit private rsa's: 10 4096 bit private RSA's in 8.51s
Doing 409 4096 bit public rsa's: 409 4096 bit public RSA's in 5.09s
OpenSSL 0.9.8 05 Jul 2005
built on: Mon Sep 19 08:36:38 2005
options:bn(64,32) md2(int) rc4(idx,int) des(ptr,cisc,4,long) aes(partial) idea(int) blowfish(idx)
compiler: bcc32 -DWIN32_LEAN_AND_MEAN -q -w-aus -w-par -w-inl  -c -tWC -tWM -DOPENSSL_SYSNAME_WIN32 -DL_ENDIAN -DDSO_WIN
32 -D_stricmp=stricmp -O2 -ff -fp -DOPENSSL_NO_RC5 -DOPENSSL_NO_MDC2 -DOPENSSL_NO_KRB5
available timing options: TIMEB HZ=1000
timing function used: ftime
                  sign    verify    sign/s verify/s
rsa 1024 bits 0.019281s 0.000969s     51.9   1032.1
rsa 2048 bits 0.123259s 0.003430s      8.1    291.6
rsa 4096 bits 0.851200s 0.012438s      1.2     80.4
C:\openssl\OUT32>_
```

質數的機率

- $\pi(x)$：所有不大於x的質數個數
- $\pi(10)=4$，$\pi(100)=25$，$\pi(1000)=168$
- 質數定理 (Prime Number Theorem)說明當x趨近於無限大時，π(x)大約等於 $\frac{x}{\ln x}$，其中ln x代表x的自然對數值

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

- 機率(150位隨機整數為質數) = $\frac{\pi\left(10^{150}\right)-\pi\left(10^{149}\right)}{9\times10^{149}}$

$$\approx\frac{\frac{10^{150}}{150\times(\ln 10)}-\frac{10^{149}}{149\times(\ln 10)}}{9\times10^{149}}=\frac{\frac{10}{9}}{150\times(\ln 10)}-\frac{\frac{1}{9}}{149\times(\ln 10)}=\frac{134}{20115\times(\ln 10)}\approx\frac{1}{346}$$

模算術(modular arithmetic)

- 兩整數a與b及一正整數n，如果 $a = b + k \times n$

成立，且k為整數，則a與b同餘(congruence)模n，亦即a與b除n所得的餘數會相同，用同餘符號(≡)

$$a \equiv b \mod n$$

- $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
- $(a - b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$
- $(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n$
- $(a \times (b+ c)) \bmod n = (((a \times b) \bmod n) + ((a \times c) \bmod n))) \bmod n$

費瑪定理(Fermat's Theorem)

- 若質數$p = 11$，小於p的正整數形成集合$\boldsymbol{Z}^*p = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$，且選取$a \in \boldsymbol{Z}^*p$，$b \in \boldsymbol{Z}^*p$，將$\boldsymbol{Z}^*p$的所有元素單獨乘某元素$a$再mod p，則會得到相同的集合。例如，將{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}的元素單獨乘2 再mod p，則得到{2, 4, 6, 8, 10, 1, 3, 5, 7, 9}= $\boldsymbol{Z}^*p$，只是與原$\boldsymbol{Z}^*p$的排列順序不同而已。若將{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}的元素單獨乘3 再mod p，則得到集合{3, 6, 9, 1, 4, 7, 10, 2, 5, 8}，還是等於$\boldsymbol{Z}^*p$。
- 所以如果$1 \le a \le p-1$，則明顯地可觀察到將$\boldsymbol{Z}^*p$內的所有元素單獨乘某元素 a再mod p，所得到的集合與原集合是相等
- $\{(1 \times a \bmod p), (2 \times a \bmod p), (3 \times a \bmod p), \cdots, ((p-1) \times a \bmod p)\} = \{1, 2, 3, .. p-1\}$

模數為11的模乘法運算

mod 11	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

費瑪定理(Fermat Theorem)

- $[(1\times a \bmod p)\times(2\times a \bmod p)\times(3\times a \bmod p)\times\cdots\times((p-1)\times a \bmod p)] \bmod p = [1\times 2\times 3\times\cdots\times(p-1)] \bmod p$
- $[(1\times a)\times(2\times a)\times(3\times a)\times\cdots\times((p-1)\times a)] \bmod p = [1\times 2\times 3\times\cdots\times(p-1)] \bmod p$
- $(p-1)!\times a^{p-1} \bmod p = (p-1)! \bmod p$
- 定理：如果p是質數且$GCD(a, p)= 1$，則
$$a^{p-1} \equiv 1 \bmod p$$
- 引理：如果 $2\le a \le p-1$ 且 $a^{p-1} \bmod p \neq 1$，則p不是質數

卡邁克爾數(Carmichael Number)

- 非質數且即使利用費瑪定理亦無法篩除
- 最小的卡邁克爾數為561=3×11×17(非質數)，我們可取a = 2, 5, 7, 13, 19, …，則2^{560} mod $561 = 5^{560}$ mod $561 = 7^{560}$ mod $561 = 1$；3^{560} mod $561 = \cdots = 1$

米勒-拉賓質數判斷法

已知: 奇數N，$N-1=2^s \times t$，t為奇數

求: N是否為質數

Step 1. 隨機挑選整數a, $2 \le a \le N-2$

Step 2. 計算$y = a^t \bmod N$

Step 3. 如果$y = 1$或$y = N-1$，則N可能為質數，結束

Step 4. for $i \leftarrow 1$ to s

Step 5. do $y \leftarrow y^2 \bmod N$

如果$y = N-1$，則go to Step 6

如果$y = 1$，則N為非質數，結束

Step 6. 如果$y = N-1$，則N可能為質數，結束

Step 7. 如果$y \neq N-1$，則N為非質數，結束

米勒-拉賓質數判斷法-2

- G. Miller（米勒）與M. Rabin（拉賓）提出的這種方法，每次誤判(把非質數誤當做質數)的機率均會小於四分之一，所以如果重複計算k次，其誤判的機會就會小於 $\left(\frac{1}{4}\right)^k$ 如當$k=8$，則誤判的機會將小於0.00001526
- 但實際上，我們還會考慮到質數的分布，當待測整數N數值夠大時，米勒-拉賓演算法的誤判率會比1/4更小
- 例如N為512位元時，進行k=6次判斷，則錯誤率便小於 $\left(\frac{1}{2}\right)^{80}$
- 如果以米勒演算法計算結果為“非質數”，那麼可再選取一個奇數，再實行米勒-拉賓演算法來進行判斷

範例: 利用米勒-拉賓質數判斷法檢查2047是否為質數

- $N=2047$，$N-1=2\times 1023$
- 若$a=2$，則$y=2^t \bmod N=2^{1023} \bmod 2047=1$，因為$y=1$，所以在步驟3進行$y$值判斷時，通過米勒-拉賓質數判斷法，即$N$可能為質數。但是若$a=3$，則$y=3^t \bmod N=3^{1023} \bmod 2047=1565$，因為$y=1565 \neq 1 \neq N-1$，所以繼續執行至步驟5，$y^2 \bmod N=1565^2 \bmod 2047=1013 \neq N-1$，因為$y \neq N-1$，符合步驟7的判斷條件，所以$N=2047$為非質數，事實上$N=2047=23\times 89$

歐拉Phi函數

- 歐拉Phi函數$\phi(n)$代表小於正整數n，但又與n互質(意即最大公因數GCD等於1)的正整數之個數。例如$n=5$，則{1,2,3,4}皆與5互質，所以$\phi(5)=4$；另外例如$n=6$，則{1,5}與6互質，所以$\phi(6)=2$，我們定義$\phi(1)=1$。
- 如果n是質數，則$\phi(n)= n-1$
- 若n是相異兩質數p與q的乘積，則$\phi(n)= \phi(p\times q)= \phi(p) \times \phi(q) = (p-1) \times (q-1)$

n	1	2	3	4	5	6	7	8	9	10	11	12
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4

歐拉定理(Euler Theorem)

- 如果GCD(a, n) = 1，則 $a^{\phi(n)} \equiv 1 \bmod n$
- 若選取$a \in Z^*_{\phi(n)}$，並將$Z^*_{\phi(n)}$的所有元素分別乘以a再mod n，則會得到相同的集合
- $Z^*_{\phi(n)}=\{R_1, R_2, R_3, \cdots, R_{\phi(n)}\}$ ，$\{(R_1 \times a \bmod n), (R_2 \times a \bmod n), (R_3 \times a \bmod n), \cdots, (R_{\phi(n)} \times a \bmod n)\} = \{R_1, R_2, R_3, \cdots, R_{\phi(n)}\}$
- $(R_1 \times a \bmod n) \times (R_2 \times a \bmod n) \times (R_3 \times a \bmod n) \times \cdots \times (R_{\phi(n)} \times a \bmod n) \equiv (R_1 \times R_2 \times R_3 \times \cdots \times R_{\phi(n)}) \bmod n$
- $(R_1 \times R_2 \times R_3 \times \cdots \times R_{\phi(n)}) \times a^{\phi(n)} \bmod n \equiv (R_1 \times R_2 \times R_3 \times \cdots \times R_{\phi(n)}) \bmod n$

RSA明文加解密能還原

- $D \times E \equiv 1 \bmod \phi(N)$，代表存在一整數$t$使得$D \times E = 1 + t \times \phi(N)$ ，所以得到下列公式：

  $M^{E\,D} \bmod N = M^{1+\,t \times \phi(N)} \bmod N = M \times (M^{\phi(N)})^t \bmod N \equiv M \times (M^{\phi(N)} \bmod N)^t \bmod N$

如果$\mathrm{GCD}(M, N) = 1$，則從歐拉定理$M^{\phi(N)} \bmod N \equiv 1$

$M^{E\,D} \bmod N \equiv M \times (1)^t \bmod N \equiv M$

- 如果$\mathrm{GCD}(M, N) \neq 1$，則在$M$是$P$的倍數或$Q$的倍數的情況下，會得到同樣的結果

RSA質數產生法

RSA產生私密金鑰質數P(或Q)的方法

一、將P最高兩位元及最低位元設定為1。

- 最低位元(least-significant bit)設為1，可確保所產生的亂數為奇數；而最高(most-significant)兩位元設為1，是為了確保P與Q乘積N的最高位元為1；若P與Q各512位元，則最高兩位元設為1代表$2^{512} > P,Q \geq 2^{511}+2^{510}+1$。那麼$N=P \times Q$，而且$N \geq (2^{511}+2^{510}+1) \times (2^{511}+2^{510}+1) > 2^{1022}+2 \times 2^{1021}+2^{1021} = 2^{1023}+2^{1021} \geq 2^{1023}$

- 所以將P與Q最高兩位元設為1時，可確保N的最高位元為1。

二、隨機產生P的其他位元。

三、檢查P是否能被小質數所整除。

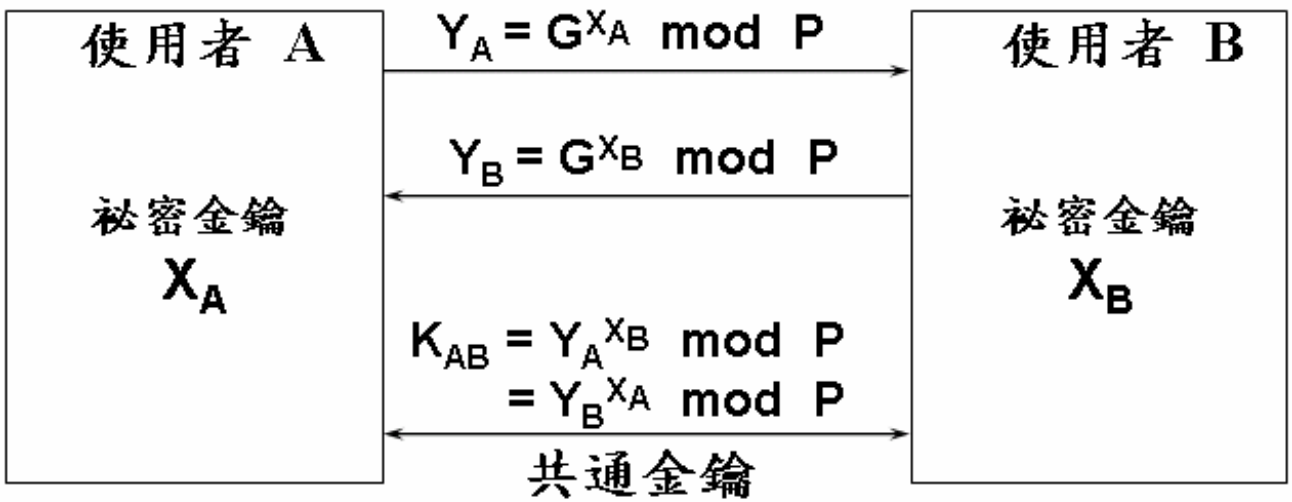
- 先以小質數3, 5, 7, 11, ...檢查P是否為這些小質數的倍數，因為這樣可以有效減少步驟4執行的次數；小質數的大小通常以一個位元組(byte)為限，也就是小於256的質數。

四、進行k次米勒-拉賓質數判斷法。

- k次執行中若有任何一次傳回“非質數”，則我們就必須回到步驟1，從新隨機挑選整數，再進行驗算。若P超過512位元，則k=8連續八次傳回P可能為質數，已可確保誤判率小於$(0.5)^{80}$。

Diffie-Hellman公開金鑰演算法

- 1976年美國史丹福大學(Stanford University)兩位學者Diffie與Hellman發明公開金鑰交換演算法（Diffie-Hellman），可用來使從未曾通信見面的雙方安全地取得共通金鑰，而這共通金鑰可做為雙方未來資料加解密或防偽之用



Diffie與Hellman公開金鑰交換的步驟

一、尋找雙方同意的一個大質數P與P的原根G，所有使用者均可用相同且公開的P與G；G為P的原根(primitive root)，代表G的1到P-1不同次方mod P會產生1到P-1之所有相異整數，不過順序可能有所不同。

$\{G \bmod P, G^2 \bmod P, G^3 \bmod P, ..., G^{P-1} \bmod P\} = \{1, 2, 3, ..., P-1\}$

二、使用者A選擇祕密金鑰正整數X_A，$2 \le X_A \le P-2$，計算

$Y_A = G^{X_A} \bmod P$，將Y_A送給欲通信的使用者B。

三、使用者B選擇祕密金鑰正整數X_B，$2 \le X_B \le P-2$，計算

$Y_B = G^{X_B} \bmod P$，將Y_B送給欲通信的使用者A。

四、使用者A利用自己的祕密金鑰整數X_A與收到的Y_B，計算

$K_{AB} = Y_B^{X_A} \bmod P = G^{X_B X_A} \bmod P$

五、使用者B利用自己的祕密金鑰整數X_B與收到的Y_A，計算出公式

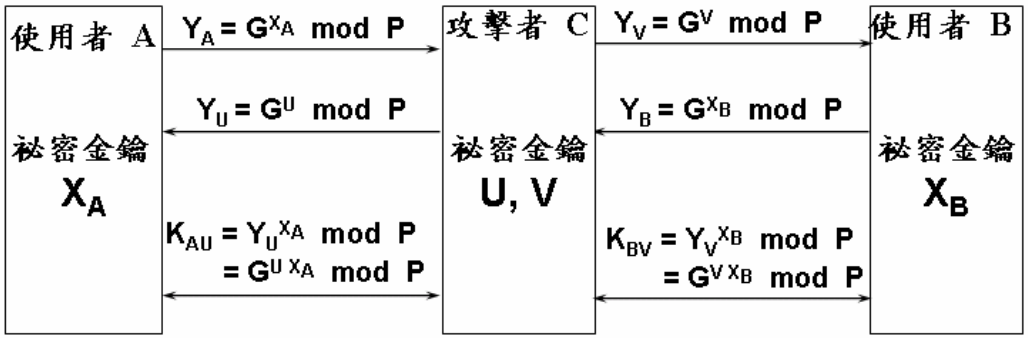
$K_{AB} = Y_A^{X_B} \bmod P = G^{X_A X_B} \bmod P$

六、雙方有了共通的金鑰$K_{AB} = G^{X_A X_B} \bmod P$。

離散對數(discrete logarithm)問題

- 如果網路中有窺視者(eavesdropper)想竊取資料，則只能取得系統公開的參數P與G以及通信過程中的兩筆資料$G^{X_A} \bmod P$與$G^{X_B} \bmod P$
- 欲從大質數P與G與$G^X \bmod P$尋找到X是很困難的，這就是所謂的離散對數(discrete logarithm)問題
- 離散對數問題困難度與RSA的大數分解類似

中間人攻擊(man-in-the-middle attack)



- 若使用者A原先要給使用者B的訊息$Y_A = G^{X_A} \bmod P$被攻擊者C收到後，竄改為$Y_V = G^{X_V} \bmod P$，這將使得B計算出的共同金鑰為$K_{VB} = Y_V^{X_B} \bmod P = G^{X_V X_B} \bmod P$
- 而攻擊者C收到B送出的$Y_B = G^{X_V} \bmod P$後，可將自己的祕密金鑰整數X_V與收到的Y_B計算出$K_{VB} = Y_B^{X_V} \bmod P = G^{X_B X_V} \bmod P$，與使用者B計算出的共同金鑰相同。
- 使用者A算出的$K_{AU} = Y_u^{X_A} \bmod P = G^{X_U X_A} \bmod P$與攻擊者C相同。
- 如此一來，使用者A與使用者B實際上都與攻擊者C進行安全通信，而非與原來約定的對方進行安全通信。攻擊者若能清楚得知A與B之間的共通金鑰，便可將收到的密文先進行解密，再重新加密後送給不知情的對方。

ElGamal公開金鑰加解密演算法

- ElGamal在1985年提出一種以離散對數為基礎的公開金鑰加解密演算法
- 採用隨機加密(randomized encryption)技術
- 意即相同的明文每次加密都會產生不同的密文。同一個明文可對應到多個不同的密文，這些不同的密文卻能解密為原來的明文
- 密文的長度為明文的兩倍

ElGamal加解密演算法-2

安全原理：離散對數問題，用大質數 P 與原根 G 與 $G^X \bmod P$ 找 X 是很困難的

金鑰產生：1. 系統參數大質數 P 與 P 的原根 G, P 和 G 為系統參數

2. 使用者隨機選擇私密金鑰正整數 X，$2 \le X \le P-2$

私密金鑰: X

3. 使用者計算 $Y = G^X \bmod P$

公開金鑰: Y

加密: 發文者取得收文者的公開金鑰 Y，隨機產生一個亂數 K，$1 \le K \le P-2$，計算密文 $C = (C_1, C_2)$

$$C_1 = G^K \bmod P$$

$$C_2 = M\,Y^K \bmod P$$

解密: 收文者使用自己的私密金鑰 X，計算明文 M

$$M = C_2\,C_1^{-X} \bmod P$$

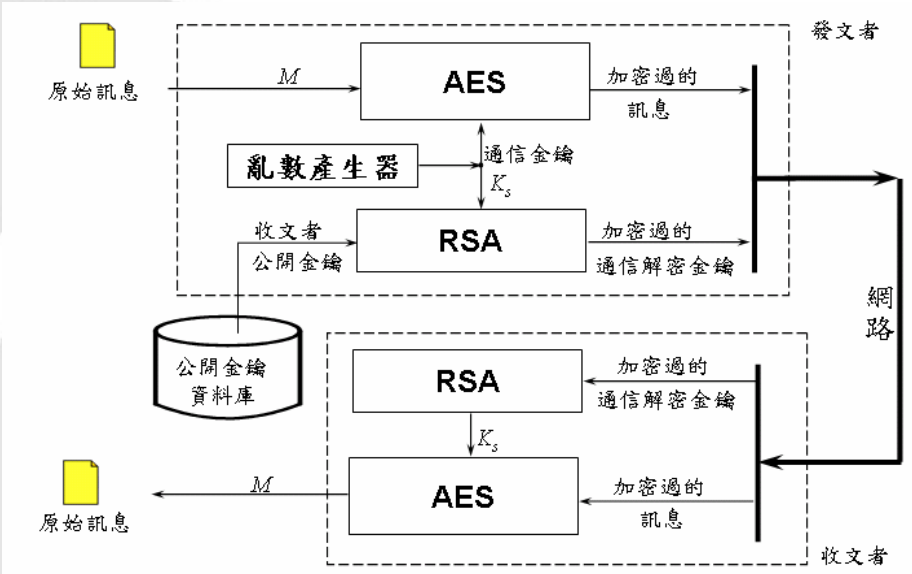
$$\begin{aligned} & C_2\,C_1^{-X} \bmod P \\ &= M\,Y^K\,(G^K)^{-X} \bmod P \\ &= M\,(G^X)^K\,(G^K)^{-X} \bmod P \\ &= M\;\; G^{XK}\,G^{-XK} \bmod P \\ &= M\;\; G^{XK}\,G^{-XK} \bmod P \\ &= M \end{aligned}$$

ElGamal加解密演算法-3

- 明文為一個小於*P*的整數，而密文為兩個小於*P*的整數，明顯地密文的長度為明文的兩倍
- 加密時須進行兩次模指數運算，乍看之下似乎要比RSA演算法慢一倍
- 然而模指數運算的基數(base)與明文無關，僅與公開金鑰*G*與*Y*有關
- 我們可利用所謂事先計算(precomputation)的技巧，先將*G*與*Y*的某些指數次方mod *P*計算儲存，以加速加密過程

- 類似RSA的公開金鑰加解密演算法可利用來進行金鑰配送
- 資料*M*可使用上一章介紹的AES演算法予以加密，但進行AES加密時，可用一個由亂數產生的加解密金鑰*Ks*，只有對方收到這個金鑰以後才能夠解密，所以這個金鑰需要安全的傳送給對方，且不能被第三者知道內容
- 我們可把這個加解密金鑰當作是RSA公開金鑰加解密演算法的明文，然後用對方的公開金鑰將其加密，即計算$Ks^E \bmod N$，再與以AES加密過的訊息一起透過網路傳送給對方，

數位信封-2



- 結合了私密金鑰密碼系統快速加解密的優點與公開金鑰密碼系統金鑰配送的優點，以達到快速又安全地資料傳送目的

一、封信(sealing): 發文者先將需傳送的訊息，利用私密金鑰密碼系統(private-key cryptosystem，如AES)的加解密金鑰予以加密，而用來進行訊息加密的加解密金鑰，則根據收文者在公開金鑰密碼系統(public-key cryptosystem，如RSA) 上，所對外公開之加密金鑰予以加密後，再與加密訊息一同傳送出去。

二、拆信(opening): 收文者先以自己在公開金鑰密碼系統上，所儲存的對外不公開之私密金鑰，解密出可以解開訊息的「解密金鑰」，再依據解密出的「解密金鑰」利用私密金鑰密碼系統將訊息恢復成未經加密前的原狀。

中國剩餘定理

- 中國古代數學家孫子的中國剩餘定理(Chinese Remainder Theorem，簡稱CRT)

已知 $\begin{cases} X \bmod m_1 \equiv v_1 \\ X \bmod m_2 \equiv v_2 \\ \quad\quad \cdots \\ X \bmod m_t \equiv v_t \end{cases}$ $\mathrm{GCD}(m_i, m_j)= 1$，且餘數 $0 \le v_i < m_i$

- $0 \le X < M$ 有唯一的解 $\quad M = \prod_{i=1}^{t} m_i$

$$X = \sum_{i=1}^{t} v_i \frac{M}{m_i} y_i \bmod M$$

$$y_i = \left(\frac{M}{m_i} \right)^{-1} \bmod m_i$$

中國剩餘定理範例

$$\begin{cases} X \bmod 5 \equiv 2 \\ X \bmod 7 \equiv 1 \\ X \bmod 11 \equiv 3 \\ X \bmod 13 \equiv 8 \end{cases}$$

求最小的正整數X

$5 \times 7 \times 11 \times 13 = 5005$

$y_1 = (5005 / 5)^{-1} \bmod 5 = (1001)^{-1} \bmod 5 = (1001 \bmod 5)^{-1} \bmod 5 = 1)^{-1} \bmod 5 = 1$

$y_2 = (5005 / 7)^{-1} \bmod 7 = (715)^{-1} \bmod 7 = (1)^{-1} \bmod 7 = 1$

$y_3 = (5005 / 11)^{-1} \bmod 11 = (455)^{-1} \bmod 11 = (4)^{-1} \bmod 11 = 3$

$y_4 = (5005 / 13)^{-1} \bmod 13 = (385)^{-1} \bmod 13 = (8)^{-1} \bmod 13 = 5$

然後利用公式（3.46）計算X

$X = 2 \times 1001 \times 1 + 1 \times 715 \times 1 + 3 \times 455 \times 3 + 8 \times 385 \times 5 \bmod 5005$

$= (2002 + 715 + 4095 + 15400) \bmod 5005 = 22212 \bmod 5005 = 2192$

用中國剩餘定理來加速RSA解密速度

- 加快$C^D \bmod N$，先用私密金鑰P, Q, D算出三個長度為模數N一半的變數D_P , D_Q, Q_P
  - $D_P = D \bmod P-1$
  - $D_Q = D \bmod Q-1$
  - $Q_P = Q^{-1} \bmod P$
- 密文C解密
  - $M_P = (C \bmod P)^{DP} \bmod P$
  - $M_Q = (C \bmod Q)^{DQ} \bmod Q$
  - $V = Q_P (M_P - M_Q) \bmod P$
  - $M = M_Q + Q \times V$
- 在最佳情況下它的解密速度會比原先公式快約4倍，而根據筆者的實作經驗至少會快三倍

Garner's Algorithm (*GA*)

- 已知: $\begin{cases} X \bmod m_1 \equiv v_1 \\ X \bmod m_2 \equiv v_2 \\ \quad\cdots \\ X \bmod m_t \equiv v_t \end{cases}$ ，GCD(m_i , m_j)= 1， 且餘數$0 \le v_i < m_i$。
- 求: X , $0 \le X < M = \prod_{i=1}^{t} m_i$

Step 1. For i from 2 to t do the following:

$C_i \leftarrow 1$.

For j from 1 to $(i-1)$ do the following:

$u \leftarrow m_j^{-1} \bmod m_i$

$C_i \leftarrow u \times C_i \bmod m_i$

Step 2. $u \leftarrow v_1$, $X \leftarrow u$

Step 3. For i from 2 to t do the following :

$u \leftarrow (v_i - X) \times C_i \bmod m_i$,

$X \leftarrow X + u \times \prod_{j=1}^{i-1} m_j$

Step 4. Return (X)

Garner演算法範例

$$\begin{cases} X \bmod 5 \equiv 2 \\ X \bmod 7 \equiv 1 \\ X \bmod 11 \equiv 3 \\ X \bmod 13 \equiv 8 \end{cases}$$

- Step 1的運算，計算出$C_2 = 3, C_3 = 6, C_4 = 5$。
- Step 3的迴圈運算，(i, u, X)分別是：(1, 2, 2), (2, 4, 22), (3, 7, 267), (4, 5, 2192)。
- 最後Step 4傳回的$X = 2192$

印度剩餘定理(IRT)

- 印度數學家Aryabhata西元499年*Aryabhatiya*
- 若將IRT跟Garner演算法做一比較，IRT較容易理解，而其效率與Garner演算法相差不大

如果正整數m_1與m_2彼此互質，意即GCD(m_1 , m_2)= 1，且餘數$0 \le v_i < m_i$，則下列同餘式組有唯一解X, $0 \le X < M = m_1 \times m_2$

$$\begin{cases} X \bmod m_1 \equiv v_1 \\ X \bmod m_2 \equiv v_2 \end{cases}$$

X = IRT $(v_1, v_2; m_1, m_2; M)$

= IRT $(0, c; m_1, m_2; M) + v_1$ ，其中$c = (v_2 - v_1) \bmod m_2$

= $Y + v_1$，其中$Y = m_1 [(c \times m_1^{-1}) \bmod m_2]$

解最多只包含一個非零餘數的同餘式組

$$\begin{cases} Y \bmod m_1 \equiv 0 \\ Y \bmod m_2 \equiv c \end{cases}$$

$Y = m_1 [(c \times m_1^{-1}) \bmod m_2]$

- 明顯地可看出Y是的m_1倍數
- $Y \bmod m_2 = m_1 [(c \times m_1^{-1}) \bmod m_2] \bmod m_2 = c \times [m_1 (m_1^{-1}) \bmod m_2] \bmod m_2 = c$

印度剩餘定理範例

- $X \bmod 3 = 1,\ X \bmod 4 = 3,\ X \bmod 5 = 3$. 用IRT求$X$最小等於多少。
- 解答：43。
- X = IRT$(1, 3, 3; 3, 4, 5; 60)$

首先求$X' = X \bmod 12$ = IRT $(1, 3; 3, 4; 12)$

$= \text{IRT}\ (0, 2; 3, 4; 12) + 1$

$= 3\ [(2 \times 3^{-1}) \bmod 4] + 1$

$= 3 \times 2 + 1 = 7$

而後X = IRT $(7, 3; 12, 5; 60)$

$= \text{IRT}\ (0, (3-7) \bmod 5; 12, 5; 60) + 7$

$= \text{IRT}\ (0, 1; 12, 5; 60) + 7$

$= 12\ [(1 \times 12^{-1}) \bmod 5\ ] + 7$

$= 12 \times 3 + 7 = 43$

t個同餘式的印度剩餘定理

$$\begin{cases} X \bmod m_1 \equiv v_1 \\ X \bmod m_2 \equiv v_2 \\ \quad\cdots \\ X \bmod m_t \equiv v_t \end{cases}$$

已知: ，$\mathrm{GCD}(m_i, m_j)= 1$，且餘數$0 \le v_i < m_i$。

求: X , $0 \le X < M$ =,$X = \mathrm{IRT}\ (v_1, v_2, \ldots, v_t; m_1, m_2, \ldots, m_t ; M)$

Step 1 : $X_1 = v_1$

Step 2 : $X_2 = \mathrm{IRT}\ (v_1 , v_2 ; m_1, m_2 ; M_2)$ $\quad M_2 = m_1 m_2$

$= \mathrm{IRT}\ (0 , |v_2 - v_1|_{m2} ; m_1 , m_2 ; M_2) + v_1$

Step 3 : $X_3 = \mathrm{IRT}\ (X_2, v_3 ; M_2, m_3 ; M_3)$ $\quad M_3 = m_1 m_2 m_3$

$= \mathrm{IRT}\ (0, |v_3 - X_2|_{m3} ; M_2, m_3 ; M_3) + v_2$

.....

Step t : $X_t = \mathrm{IRT}\ (X_{t-1}, v_t ; M_{t-1}, m_t ; M_t)$ $\quad M_t = M$

$= \mathrm{IRT}\ (0, |v_t - X_{t-1}|_{mt} ; M_{t-1}, m_t ; M_t) + v_t$

t個同餘式的印度剩餘定理演算法

已知: $\begin{cases} X \bmod m_1 \equiv v_1 \\ X \bmod m_2 \equiv v_2 \\ \quad\cdots \\ X \bmod m_t \equiv v_t \end{cases}$ ，$\mathrm{GCD}(m_i, m_j)= 1$ ，

且餘數$0 \le v_i < m_i$

求: X , $0 \le X < M = \prod_{i=1}^{t} m_i$

Step 1. $N_1 \leftarrow 1$, $X_1 \leftarrow v_1$

Step 2. For i from 2 to t do the following:

$N_i \leftarrow N_{i-1} \times m_{i-1}$

$C_i \leftarrow N_i^{-1} \bmod m_i$ (also denote $\left| N_i^{-1} \right| m_i$))

$U_i \leftarrow [(v_i - X_{i-1}) \times C_i] \bmod m_i$

$X_i \leftarrow X_{i-1} + U_i \times N_i$

印度剩餘定理範例2

- 求X = IRT (*2, 1, 3, 8 ; 5, 7, 11, 13 ; 5005)*

i	N_i	$N_i \bmod m_i$	C_i	U_i	X_i
1	1	--	--	--	2
2	5	5	$\lvert 5^{-1}\rvert_7 = 3$	$\lvert(1-2)\times 3\rvert_7 = 4$	$2+ 4\times 5 = 22$
3	$5\times 7 = 35$	$\lvert 35\rvert_{11} = 2$	$\lvert 2^{-1}\rvert_{11} = 6$	$\lvert(3-22)\times 6\rvert_{11} = 7$	$22+7\times 35 = 267$
4	$35\times 11 = 385$	$\lvert 385\rvert_{13} = 8$	$\lvert 8^{-1}\rvert_{13} = 5$	$\lvert(8-267)\times 5\rvert_{13} = 5$	$267+5\times 385=2192$

參考資料

- D.M. Bressound, *Factorization and primality testing*, Springer-Verlag, 1989.
- A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press Series on Discrete Mathematics and Its Applications, 1996.
- The Great Internet Mersenne Prime Search (GIMPS), http://www.mersenne.org/
- B. Kaliski and M. Robshaw, "The Secure Use of RSA," *Cryptobytes*, Vol. 1, No. 3, Autumn 1995, ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto1n3.pdf
- D.E. Knuth, *The Art of Computer Programming* – *Seminumerical Algorithms*, Vol. 2, 3rd edition, Addison-Wesley, 1998.
- T.R.N. Rao and Chung-Huang Yang, "Aryabhata Remainder Theorem: Relevance to Public-Key Crypto-algorithms," *Circuits, Systems & Signal Processing*, http://crypto.nknu.edu.tw/publications/2006CSSP.pdf
- K.H. Rosen, *Elementary Number Theory and Its Applications*, Addison-Wesley, 3rd edition, 1993.
- RSA Laboratories, The RSA Challenge Numbers, http://www.rsasecurity.com/rsalabs/node.asp?id=2093.