因數分解與 RSA 密碼系統

官大智 教授*

August 25, 2003

Abstract

因數分解是一個很古老的數學問題, 它同時也是近代密碼學的重要理論基礎. 因此, 研究因數分解不僅是一個很有趣的數學問題, 也具有極高的應用價值. 本文的主要目的是向大家介紹因數分解的各種方法, 以及這些方法和近代密碼學之間的關係.

在 1978 年, 三位密碼學家 Rivest, Sharmir, 和 Adleman 提出 RSA 密碼系統. 這個密碼系統的加密金鑰和解密金鑰不相同, 而且由加密金鑰推導出解密金鑰在計算上是一個很困難的工作, 因此加密金鑰可以公開, 稱爲"公開金鑰密碼系統." 這種方式可以讓通信的雙方不需要事先擁有秘密的金鑰, 非常適合現代的電腦網路系統的運作環境.

簡單的說, RSA 密碼系統的運作是這樣的. 首先, 資訊必須轉換成數字, 假設是正整數 m. 這一點很容易做到, 因爲存在電腦中的資料都是 0 和 1 的信號, 只要將資料切割成固定長度的位元, 例如: 512 或 1024 位元, 然後將它們看成一個正整數 m 就可以了. 然後選擇一個比 m 還要大的整數 n, 以及加密用的金鑰 e. 這樣就可以將明文 m 轉換成密文 c:

$$c = m^e \bmod n$$

其中, m^e 表示 m 的 e 次方, $\bmod n$ 表示除以 n 的餘數. 在實際應用的時候, n 和 e 都是事先選定的, 它們必須符合特定的條件, 不可任意選擇. 本文的主要目的就是說明這些參數的選擇與因數分解的關係.

現在用很小的例子來說明. 假設 $n = 143$, $e = 103$.

$$m = 2, \quad c = 2^{103} \bmod 143 = 63 (= 101412048018258352119736256430 08 \bmod 143)$$

$$m = 3, \quad c = 3^{103} \bmod 143 = 16$$

$$m = 4, \quad c = 4^{103} \bmod 143 = 108$$

*國立中山大學資訊工程學系

$m = 5, \quad c = 5^{103} \bmod 143 = 125$

$m = 6, \quad c = 6^{103} \bmod 143 = 7$

$m = 7, \quad c = 7^{103} \bmod 143 = 123$

由以上的例子可以察覺到, 將明文 m 自乘 e 次之後, 再求除以n 的餘數, 則此餘數將會很沒有規則. 也就是說, 給定一個數 c, 要反求 m, 使的其密文恰好是 c, 是一件很困難的計算工作. 大家也許會問從 0 到 $n-1$ 一個個的去試, 不就可以得到答案了嗎? 當 n 很小的時候當然可以這樣做, 但是當 n 很大的時候, 一個個的去試就有麻煩了. 科學家認爲從太陽系誕生到現在大約是 46 億年, 而且認爲太陽系是在生命的中期, 也就是說我們剩下的時間還有 46 億年, 這個時間小於 10^{18} 秒. 以每秒鐘可以試一百萬個不同的 m 的計算機來計算, 10^{18} 秒的時間大概可以測試到 10^{24}，也就是才 24 位數！ 通常在 RSA 密碼系統中所使用的 n 有 150 多位數，甚至更大.

在 RSA 密碼統中, 合法的收信者收到密文之後, 如何將密文轉換成明文呢? 這要用到下列公式：

$$\text{若 } a \text{ 與 } n \text{ 互質, 則 } a^{\phi(n)} \equiv 1 \pmod{n}$$

其中 $\phi(n)$ 是從 1 到 $n-1$ 的整數中與 n 互質的整數的個數. (兩個整數互質, 就是此兩整數的最大公因數是 1 的意思.) 舉例來說 $\phi(6) = 2$, 因爲從 1 到 5 的 5 個整數中, 只有兩個整數, 1 和 5, 與 6 互質. $\phi(7) = 6$, 因爲從 1 到 6 的 6 個整數中, 每一個整數都與 7 互質.

有了 $\phi(n)$ 之後, 就可以由密文計算出明文, 而不用一一去嘗試. 方法是先計算出一個數 d, 使得:

$$d \cdot e \equiv 1 \pmod{\phi(n)}$$

也就是說 $d \cdot e = k \cdot \phi(n) + 1$. 至此, 大家就可以看出只要將密文 c 自乘 d 次, 再求除以 n 的餘數, 就可以的到明文 m:

$$\begin{aligned} c^d \bmod n &= (m^e)^d \bmod n \\ &= m^{(ed)} \bmod n \\ &= m^{(k\phi(n)+1)} \bmod n \\ &= m^{(k\phi(n))} m \bmod n \\ &= ((m^{\phi(n)})^k) m \bmod n \\ &= m \end{aligned}$$

例如: $n = 143$, $e = 103$ 時, $\phi(n) = 120$, 計算得 $d = 7$, 因爲 $(7)(103) \bmod 120 =$

$721 \bmod 120 = 1$.

$$
\begin{aligned}
c = 63, &\quad m = 63^7 \bmod 143 = 2 \ (= 3938980639167 \bmod 143) \\
c = 16, &\quad m = 16^7 \bmod 143 = 3 \\
c = 108, &\quad m = 108^7 \bmod 143 = 4 \\
c = 125, &\quad m = 125^7 \bmod 143 = 5 \\
c = 7, &\quad m = 7^7 \bmod 143 = 6 \\
c = 123, &\quad m = 123^7 \bmod 143 = 7
\end{aligned}
$$

我們在回頭看看 $d = 7$ 是如何計算得到的. 因爲 $d \cdot e \equiv 1 \pmod{\phi(n)}$, 我們稱 d 和 e 互爲反元素. 由此可知, e 的值不可以隨便取, 它的值要與 $\phi(n)$ 互質才行, 否則就找不到反元素 d 了. 若 e 與 $\phi(n)$ 互質, 則由最大公因數表現定理可知: 存在兩個整數 x, y 使得

$$e \cdot x + \phi(n) \cdot y = \gcd(e, \phi(n)) = 1$$

將上面等式的兩邊同時做 $\bmod \phi(n)$ 就可以得到 $e \cdot x \equiv 1 \pmod{\phi(n)}$. 因此, e 的反元素就是 x, 它的值可以用輾轉相除法求得.

最後要說明 $\phi(n)$ 的值要如何計算. 依照定義, $\phi(n)$ 就是從 1 到 $n-1$ 之間的整數與 n 互質的個數. 但是直接依照定義去檢查這些整數是否與 n 互質, 當 n 很大的時候是不實際的. 假設知道 n 的因數分解,

$$n = \prod_{k=1}^{r} p_k^{e_k}$$

則這些質數 p_k, $k = 1, 2, \ldots, r$, 的整倍數都與 n 不互質; 反之, 則與 n 互質. 由此可以推算得:

$$\phi(n) = \prod_{k=1}^{r} p_k^{e_k - 1}(p_k - 1)$$

除了上面的等式之外, 目前不知道有任何其他的方法可以快速地求得 $\phi(n)$ 之值. 這點對 RSA 密碼系統的安全性是非常重要的. 因爲只要能夠計算出 $\phi(n)$, 就可以用加密金鑰 e 求得解密金鑰 d. 因此, RSA 密碼系統的安全就和因數分解建立起密不可分的關係.

在說明因數分解的方法之前, 我們再談 RSA 密碼系統的一個特性. RSA 密碼系統有一個很大的好處, 那就是加密和解密的金鑰可以不同. 而且, 由其中一個金鑰不容易求得另外一個金鑰, 除非你會分解很大的數的因數. 因此, 加密的金鑰可以公開, 此系統又稱爲"公開金鑰" 密碼系統. 在公開金鑰密碼系統中, 要跟對方通信的人並不需要事先取得共用的秘密金鑰, 這點對電腦網路的應用非常重要, 因爲在電腦網路上是無法傳送秘密金鑰的.

現在來說明因數分解和 RSA 密碼系統中 n 的值的選擇. 大整數的因數分解到底有多困難? 在 RSA 密碼被提出來的當初, 設計者提出一系列的大整數, 來挑戰分解大整數的速度. 其中一個 129 位數的整數爲:

$$114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541$$

此數在提出 17 年之後, 終於被分解出來. 分解這個 129 位數使用了世界各地的 1600 餘部電腦工作站, 一共花了八個多月才計算出答案. 它是下面二數的乘積:

$$34905295108476509491478496199038981334177646384933878439908205773276913299326670954996198819083446141317764296799294253979828853$$

由上面的事實可以推得, 既使計算機的計算速度以每兩年左右的時間就增快一倍, 要分解 200 位數的整數在目前的技術而言仍然是一件很困難的工作. 但是對特殊的 n 有一些因數分解的方法可以很快地做因數分解, 這一點會影響 n 的選擇.

現在我們就來談如何選擇適當的 n, 使得 RSA 密碼系統具有足夠的安全性. 首先, n 不可以是一個質數. 如果 n 是一個質數, 則 $\phi(n) = n - 1$. n 也不可以是一個質數的幾次方, 若 $n = p^s$, 則可以用數值方法解 $x^s - n = 0$ 的實根, 而得到 p. 所以, n 的因數必須至少包含兩個不同的質數.

分解 n 的因數最直接的方法是一一測試介於 2 和 $\sqrt{n}$ 之間的整數 是否可以整除 n. 但是這個方法在 n 很大的時候是很沒有效率的, 因爲要測試的除數實在太多了. 事實上, 並不需要測試介於 2 和 $\sqrt{n}$ 之間的所有整數, 只需要測試介於 2 和 $\sqrt{n}$ 之間的質數是否可以整除 n 就可以了. 即使如此, 當 n 的值很大的時候, 如何產生或儲存這些質數也是一個困難的問題. 雖然如此, 如果 n 是許多小的質數的乘積的時候, 這個方法就可以成功的分解 n. 這就是爲什麼 n 通常是取兩個很大質數的乘積的原因.

令 p 和 q 爲兩個很大的質數, 且 $n = p \cdot q$. 雖然不知道 p 和 q 的值, 但是當它們的值很相近的時候, 就能夠有機會分解 n. 因爲 p 和 q 的值很相近, 所以它們的值接近 $x = [\sqrt{n}]$. 令 $p = x - a$, $q = x + b$, 其中 a 和 b 都是非負的整數, 則:

$$\begin{aligned} n &= (x-a)(x+b) \\ &= x^2 + (b-a)x - ab \\ &= x^2 + (b-a-1)x + (x-ab)x^0 \end{aligned}$$

若 $0 \le b-a-1 < x$, 且 $0 \le x-ab < x$, 則 $n = x^2 + (b-a-1)x + (x-ab)x^0$ 就是將 n 表示成 x 進位的意思. 給定 n 和 x 的值之後, 就很容易求得 α 和 β 使得 $n = x^2 + \alpha x + \beta$. 有了 α 和 β 就可以解下列方程式求出 a 和 b 的值.

$$\begin{aligned} b-a &= \alpha+1 \\ ab &= x-\beta \end{aligned}$$

現在以一個很小的例子來說明這個因數分解的方法. 假設 $n = 187$. $x = [\sqrt{187}] = 13$. $187 = (13)^2 + 1(13) + 5$, 因此, $\alpha = 1$, $\beta = 5$. 解方程式

$$\begin{aligned} b-a &= 1+1 \\ ab &= 13-5 \end{aligned}$$

得 $a = 2, b = 4$. 所以, $p = 13-2 = 11$, $q = 13+4 = 17$.

以上的方法要能成功分解 n 的因數, a 和 b 的值必須滿足 $0 \le b-a-1 < x$, 且 $0 \le x-ab < x$. 第一個條件是說 $b > a$, 且他們的差不可超過 x. 第二個條件是說 $0 < ab \le x$. 假設 a 和 b 的值很相近, 則它們的值接近 $\sqrt{x}$. 也就是說 p 和 q 的前面最高位元開始算起, 若有一半位元數都相同時, n 就可以被上述的方法分解成功. 也就是說 p 和 q 的差至少要在 $n^{\frac{1}{4}}$ 以上才能使 RSA 密碼系統具有一定的安全性.

這些條件是否就可以保證 n 不容易被分解了呢? 不是的, 還有許多分解因數的方法會影響到 n 的選擇. 在密碼學家開始研究 RSA 系統的初期, $p-1$ 和 $p+1$ 演算法是分解因數最有效的演算法. 若 $p-1, p+1, q-1, q+1$ 只包含很小的因數, 則這些演算法可以成功的分解 n. 因此, 許多密碼學家在當時都認爲用來產生RSA密碼系統金鑰的兩個質數 p 和 q 都必須是強質數, 才能抵擋 $p-1$ 和 $p+1$分解因數的攻擊. 包含 RSA 的原創者在提出此密碼系統時也認爲如此.

Pollard 於 1974 年提出 $p-1$ 因數分解演算法. 此法所欲分解的整數 n 必須包含至少兩個質因數. 這點剛好與 RSA 密碼系統所使用的 n 相符合. 此方法首先在 1 與 $n-1$ 之間任意選擇一個整數 a. 若 a 與 n 不互質, 則其公因數就是一個 n 的因數. 反之, 若 a 與 n 互質, 則計算

$$x = a^m \bmod n,$$

其中 m 是 $p-1$ 的倍數, 而 p 是 n 的某個質因數. 但是 p 是未知的, 如何求得 m?

設 t 爲質數, k 爲正整數, 我們稱 t^k 爲質冪 (prime power). 若 t^k 可以整除 $p-1$, 則稱 t^k 爲 $p-1$ 的質冪因數 (prime-power factor). Pollard 的 $p-1$ 演算法先訂出一個上界 B, 其值通常在 10^6 左右, 然後令

$$m = \text{所有小於 } B \text{ 的質冪的乘積}$$

精確的說, 對每一個質數 t 乘上小於 B 的最大質冪即可. 假設所有 $p-1$ 的質冪因數均不超過 B, 則 m 必定是 $p-1$ 的倍數. 最後, 計算

$$d = \gcd(x-1, n)$$

若 $d > 1$, 則 d 爲 n 的一個因數, 否則失敗. 但是, 若 $x = 1$, 且 m 是偶數, 則可將 m 除以 2, 在重新計算 $x = a^m \bmod n$ 試試看. 若 $p-1$ 或 $q-1$ 的質冪因數都小於 B, Bach 等人證明此法有 0.5 的機率可以選對一個 a 使得 n 成功的被分解.

此法所以有效, 是因爲 $p-1$ 正好是乘法群 Z_p^* 的 order. 在計算 $x = a^m \bmod n$ 的過程中，雖然是用 $\bmod n$, 但是根據中國餘數定理, 若 $n = p \cdot q$, 則這些結果對於 $\bmod p$ 仍然是對的. 又因爲

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

所以 $p|(x-1)$, 也是說 若 $x \neq 1$, $d = \gcd(x-1, n)$ 就是 n 的因數.

Williams 於 1982 年提出一個類似的方法分解因數, 稱爲 $p+1$ 法. 它用 Lucas sequences, 取代計算 $x = a^m \bmod n$. 也就是說, 用有限場 $GF^*(p^2)$ 取代 Z_p^*. $GF^*(p^2)$ 的 order 爲 $p+1$. 因此, 假設所有 $p+1$ 或 $q+1$ 的質冪因數均很小, 則 n 的因數就有可能被求出來.

到了1985年, Hendrik 和 Lenstra 提出一個新的因數分解演算法. 此法是根據橢圓曲線上的點

$$E(a,b) = \{(x,y,z) | y^2 z \equiv x^3 + axz^2 + bz^3,\ 4a^4 + 27b^2 \neq 0 \pmod n\}$$

所定義的交換群來運算的. 這個交換群的 order, $|E(a,b)|$, 不僅和 p 的值有關, 也和 a 與 b 的值有關. 若 a 和 b 都是 Z_p 中的元素, 則

$$(p+1-2\sqrt{p}) \leq |E(a,b)| \leq (p+1+2\sqrt{p})$$

當 a 和 b 的值隨機選定之後, $|E(a,b)|$ 的值大約可介於上述範圍的任一值. 這個演算法更高明的地方是它可以選定不同的 a 和 b, 並且可以平行的計算. 如此一來, 要求 $p-1$ 或 $p+1$ 要有很大的質冪因數就沒有多大意義了. 因爲只對 $p-1$ 或 $p+1$ 去保護, 就像只關上兩扇門, 而其他更多的門卻仍然是開著的.

然而, 這並不表示 RSA 密碼系統是無法做到安全的. 由於這些演算法仍然需要大量的計算, 而且增加 n 的位元數, 所需要的計算量並不只是線性的增加. 因此, 我們只要增加 n 的位元數, 就可以使得 RSA 密碼系統具有一定的安全性.

以上的因數分解演算法都是針對 n 的因數 p 和 q 的特性來設計. 另外有一類的分解因數演算法只和 n 的大小有關, 而與 p 和 q 的特性無關. 例如: 二次篩選法 (quadratic seive)

就是一個典型的例子. 此法設法找出一對 x 和 y, 使得

$$\begin{aligned} x &\not\equiv \pm y \pmod{n} \\ x^2 &\equiv y^2 \pmod{n} \end{aligned}$$

由 $x^2 \equiv y^2 \pmod{n}$ 可以推得 $x^2 - y^2 \equiv 0 \pmod{n}$, 也就是說

$$n | (x+y)(x-y)$$

又由於 $x \not\equiv \pm y \pmod{n}$, 可以推得 $n \nmid (x+y)$, $n \nmid (x-y)$. 因此, $x+y$ 和 $x-y$ 各包含一個 n 的因數. 也就是說, $\gcd(n, x+y)$ 和 $\gcd(n, x-y)$ 就是 n 的兩個因數.

如何求得一組 x 和 y, 使得 $x \not\equiv \pm y \pmod{n}$ 且 $x^2 \equiv y^2 \pmod{n}$ 呢? 二次篩選法的策略是: 從 $w = [\sqrt{n}] + 1$ 開始, 一一測試 $w^2 \bmod n$ 是否可以被分解成爲 事先定好的一組質數 $\{p_1, p_2, \ldots, p_s\}$ 的乘積. 如果可以, 就得到一個 "好的" 等式:

$$w^2 \equiv \prod_{k=1}^{s} p_k^{e_k}$$

收集到比 s 還要多的這些等式之後, 就可以組成另一個等式其左邊都是某些數平方的乘積, 右邊都是 $p_1, p_2, \ldots, p_s$ 這些質數偶數次方的乘積. 兩邊分別化簡就可以得到 $x^2 \equiv y^2 \pmod{n}$. 若 $x \neq \pm y$, 就可以分解 n.

現在以一個很小的例子來說明二次篩選法。 假設 $n = 33221$. 先選定一組質數, $B = \{2, 3, 5, 7\}$. 從大於 $\sqrt{33221}$, 即 $w = 183$ 開始, 一一測試 $w^2 \bmod 33221$ 是否可以分解成質數 2, 3, 5, 7 的乘積. 以下是用程式計算的結果.

$$\begin{aligned} 189^2 &\equiv 2^2 5^4 \\ 378^2 &\equiv 2^3 3^1 7^2 \\ 567^2 &\equiv 2^2 3^2 5^4 \\ 682^2 &\equiv 2^1 3^1 5^1 \\ 818^2 &\equiv 2^5 3^1 7^2 \\ 835^2 &\equiv 3^8 5^1 \\ 845^2 &\equiv 2^{14} \\ 983^2 &\equiv 2^6 3^2 5^1 \\ 1169^2 &\equiv 2^2 3^2 5^3 \\ 1223^2 &\equiv 2^4 7^2 \end{aligned}$$

$$\begin{aligned} 1277^2 &\equiv 2^3 3^3 7^2 \\ 1337^2 &\equiv 2^3 3^3 \\ &\vdots \end{aligned}$$

由 $189^2 \equiv 2^2 5^4$ 可得

$$189^2 \equiv 2^2 5^4 \equiv 50^2$$

因此, $189^2 - 50^2 \equiv 0 \bmod 33221$, 也就是說 $(189-50)(189+50)$ 可以被 33221 整除, 亦即 $(139)(239) = k(33221)$, 可得 $33221 = 139 \times 239$. 讀者很容易發現用其它式子, 例如: 第 2 式; 第 10 式組合第 11 式, 等等, 都可以分解 $n = 33221$.

有了這個因數分解的演算法, 是不是表示 RSA 密碼系統不安全了呢? 也不是的. 以目前的計算技術而言, 這些因數分解法所能分解的整數遠小於 RSA 所使用的公開金鑰 n 的大小. 根據 Montgomery, Silverman, Wagstaff 等人的研究顯示, 要分解 1024 位元的整數仍然相當困難. 根據推算，要分解 1024 位元的整數 n, 得到 512 位元的因數 p 或 q, 必需要有大約一兆個 MIPS R10000 的計算機, 執行大約 3.75×10^{10} 年才有 0.63 的成功機率. 這個時間大約是宇宙生命的 2 倍長.

由以上的介紹，可以知道因數分解不但在理論上有意義, 在密碼學上也很有價值. 是一個值得研究的領域.