使用尤拉函數的倍數分解因數

楊伏夷
朝陽科技大學資訊工程系
yangfy@cyut.edu.tw

江承軒
朝陽科技大學資訊工程系
icemask@gmail.com

摘要

經由研究顯示，若合成數 n 為尤拉函數 $\phi(n)$ 的倍數，因數分解該合成數的時間複雜度是 $O(|g(n)||n|^4M(|n|))$。以現今密碼系統的安全參數而言，$O(|g(n)||n|^4M(|n|))$表示時間複雜度的上限大約是需要 2^{77} 個執行步驟，這樣一個不小的數據，以目前的科技設備，到底需要多少時間來完成，令人期待。本篇文章實作利用 $\phi(n)$ 的倍數對 n 做因數分解，實際量測耗費的時間，並且更進一步證實因數分解的時間與 $\phi(n)$ 倍數的位元長度呈現線性關係，時間複雜度可以由原先的 $O(|g(n)||n|^4M(|n|))$，重新估算表示，調降為 $O(|g(n)||n|^4)$。

關鍵詞：因數分解、時間複雜度、尤拉函數。

Abstract

Given the multiple of Euler's phi function $\phi(n)$, the time complexity of factoring a composite number n is $O(|g(n)||n|^4M(|n|))$. Based on security parameters in the current cryptographic system, $O(|g(n)||n|^4M(|n|))$ means the upper bound of time complexity takes about 2^{77} steps. It is quite a large number, and we are interested in the time it actually takes. We use the multiple of $\phi(n)$ to factorize n and measure the time consumed. Furthermore, we prove the linear relationship between the factorization time and bit length, and that it is possible to reduce the time complexity $O(|g(n)||n|^4M(|n|))$.

Keywords: Factoring, Time Complexity, Euler's Phi Function.

1. 簡介

RSA 公開金鑰密碼系統 [4] 廣泛應用在身分認證、電子簽章、秘密通訊等領域，其公開金鑰 (e, n)，私密金鑰 d，參數簡述如下：p 和 q 是兩個大質數； 合成數 n 是兩大質數的乘積，$n = pq$；尤拉函數(Euler's phi function) $\phi(n) = (p - 1)(q - 1)$；e 是隨意挑選的整數，符合 $1 < e < \phi(n)$與 $\gcd(e, \phi(n)) = 1$；$d = 1/e \bmod \phi(n)$。RSA 公開金鑰密碼系統的安全性是基於 RSA 數學難題的假設：雖然知道公開金鑰(e, n)，但仍然無法據此得知私密金鑰 d。

如果在已知公開金鑰(e, n)的情形下，可以計算出私密金鑰 d，則可以知道$\phi(n)$的倍數，i.e. $g(n) = (de - 1) = r\phi(n)$，其中 r 是整數並且 $1 \le r$。假設 $g(n)$是$\phi(n)$的倍數，$|g(n)|$與$|n|$分別表示字串 $g(n)$與 n 的位元長度，兩者的位元長度呈現$|g(n)|= O(|n|^k)$的關係，其中 k 是常數，根據文獻[3]的理論，當 $g(n)$已知時，合成數 n 的因數分解可以在 $O(|g(n)||n|^4M(|n|))$或者 $O(|n|^{4+k}M(|n|))$的時間複雜度內為之，其中 $M(|n|) = O(|n| \log |n| \log\log |n|)$。

由上述可知，若存在"***有效率的演算法***"來解答 RSA 數學難題，則可取得$\phi(n)$的倍數，i.e. $g(n)$，進而將合成數 n 因數分解；換句話說，合成數 n 的因數分解，可以轉換(reduce to)為求取 $g(n)$，i.e. "*prime factorization*" $\le_p g(n)$。由於因數分解歸屬於數學難題，目前尚未發現有效率的因數分解演算法，因此反證尚未找到有效率的解答方案來求取 $g(n)$，這也是 RSA 公開金鑰密碼系統安全性的基礎。文獻[5, 6]也是植基於相同安全基礎的密碼協定。

上述"***有效率的演算法***"表示演算法的時間複雜度與位元長度為多項式關係，$O(|g(n)||n|^4M(|n|))$屬於"***有效率的演算法***"。若 $|g(n)|= |n|^2$，則 $O(|g(n)||n|^4M(|n|)) \approx O(|n|^7)$，在現今科技與密碼系統的安全參數要求之下[2, 7]，$|n| \approx 2432 \approx 2^{11}$，$O(|n|^7)$時間複雜度的上限大約是需要 2^{77} 個執行步驟，這是一個不小的數據，我們不免好奇執行時間到底是多久? 本篇文章中，我們將利用實作來實際測量 $g(n)$的位元長度與轉換時間(i. e.分解因數的時間)的關係，從而證明使用 $g(n)$來因數分解合成數 n 是容易的，當合成數 n 的位元長度確定時，因數分解合成數 n 的時間複雜度可以更簡潔的表示成 $O(|r|)$。

2. 實驗方法

本篇文章的實驗是參考文獻[1, 3]的演算法，安排實作流程如下：

演算法 1.

i. 隨意產生兩大質數 p 與 q，且$|p| = |q|$，例如$|p| = |q|$= 512 位元；計算 $n = p \cdot q$ 以及$\phi(n) = (p - 1)(q - 1)$之值，將 n、p、q、與$\phi(n)$儲存備用。

ii. 隨意產生亂數 $1 < r$，其字串編碼為預定的長度，例如$|r| = 100$。

iii. $g(n) = r \cdot \phi(n)$，故 $g(n)$是$\phi(n)$的倍數。將 $g(n)$整理成 $g(n) = 2^t \cdot k$，其中 k 是奇數，t 則是正整數。

iv. 隨意產生亂數 a，$1 < a < n - 1$。若測試 $\gcd(a, n) \neq 1$ 成立，因數分解完成，記錄完成的時間，並且停止計算；否則，對每一個 s 值，$0 \le s \le t$，測試 $\gcd((a^{k \cdot 2^s} \bmod n) - 1, n) \neq 1$，成立則因數分解完成，記錄完成的時間，並且停止計算。

v. 重新回到步驟 iv.。

步驟 i 產生固定長度的合成數 n，步驟 ii 和步驟 iii 製造指定長度的$\phi(n)$倍數，步驟 iv 執行因數分解並記錄時間，當步驟 iv 未能解出合成數 n 的因子時，步驟 v 導引流程回到步驟 iv，重新選擇亂數 a。

由於亂數 a 和 r 是隨意產生，故因數分解完成的時間也將有所不同，故對於每一個預定的編碼長度而言，i.e. $|r|$和$|n|$，我們將重複若干次，統計並計算其平均值。

3. 實驗結果

本文的實驗環境如下:

- CPU : Intel P8400 2.26GHz 2.27GHz
- RAM: 4G
- OS: Windows 7
- Program Language: Java 1.6
- IDE: Netbeans 6.8

在$|n|$ = 1024 的條件之下，轉換時間(平均時間)與$|r|$的關係如表 1 與圖 1 所示；當$|n|$ = 2048 時，表現如表 2 與圖 2。可以由圖 1 與圖 2 的數據中發現，r 的位元長度並不會使因數分解的時間大幅增加，其時間複雜度並非為指數成長，而是線性成長，符合 $O(|r|)$。

4. 結論

由實驗可以發現，當 n 是兩個質數的乘積，質數的位元長度相等，若$\phi(n)$的倍數已知，則將 n 因數分解是非常容易的，並且可從實驗結果得知，分解因數的時間和 r 的位元長度是線性相關，i.e. $O(|r|)$。

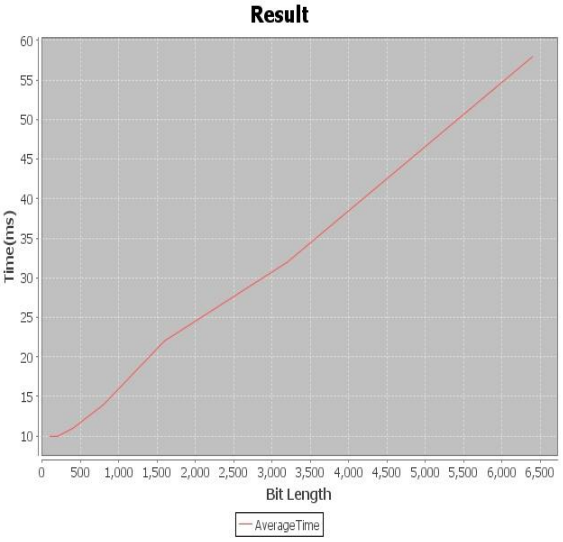

圖 1　r 的位元長度和轉換時間關係圖 ($|n|$ = 1024)

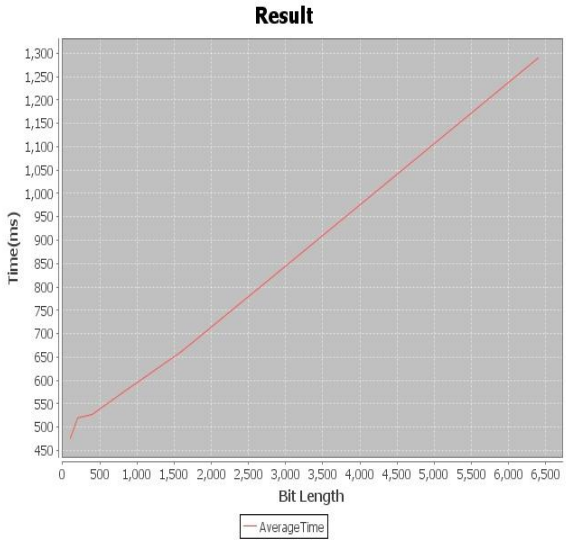

圖 2　r 的位元長度和轉換時間關係圖 ($|n|$ = 2048)

文獻[3]估計，$g(n)$已知時，分解合成數 n 的時間複雜度是 $O(|g(n)||n|^4 M(|n|))$，在$|r| = 2^{11}$ 與$|n| = 2^{11}$ 的條件之下，其上限大約是 2^{77} 個執行步驟，由圖 2 得知，耗時 730 ms 就可完成分解因數。

在第四等級(level 4)的安全保護考量下[2, 7]，對稱式加密演算法的金鑰長度是 80 位元，非對稱式加密演算法的模數位元長度是 1248 位元，時間複雜度的上限大約是 2^{80} 個執行步驟；換句話說，若時間複雜度的上限是 2^{77} 個執行步驟，則不可能僅僅花費 730 ms 就可完成分解因數。因此，是否意味著時間複雜度 $O(|g(n)||n|^4 M(|n|))$有下降的可能，討論如下。

若分析實作的流程，i.e. 演算法 1.，計算量

集中在步驟 iv，依文獻[1]的定理 8.3.8，隨意選擇亂數 a，因數分解合成數 n 的機率是 1/2，使用 Extended Euclidean Algorithm 找尋 x 與 y 的最大公約數，i.e. gcd(x, y)，時間複雜度是 $O(|x||y|)$；計算模指數乘法 $x^y \bmod n$，時間複雜度是 $O(|x||y||n|)$，因此演算法 1 的時間複雜度是 $O(|g(n)||n|^4)$，的確少了一個級數，i.e. $M(|n|)$。

表 1 位元長度和時間關係表 ($|n| = 1024$)

r's bit length	Average time (ms)
100	10
200	10
400	11
800	14
1600	22
3200	32
6400	58

表 2 位元長度和時間關係表 ($|n| = 2048$)

r's bit length	Average time (ms)
100	477
200	519
400	527
800	572
1600	662
3200	871
6400	1291

致謝

作者感謝審稿委員們寶貴的評論與建議。本研究承蒙國科會部份經費支持，謹此致謝，計畫編號：NSC 98-2221-E-324-019。

參考文獻

[1] Buchmann, J. A., "Introduction to cryptography," ***2nd ed.***, Springer, 2004.

[2] Lenstra, A. and Verheul, E., "Selecting cryptographic key sizes," ***The Third International Workshop on Practice and Theory in Public Key Cryptography* (PKC2000)**, LNCS 1751, pp. 446-465, 2000.

[3] Miller, G. L., "Riemann's Hypothesis and tests for primality," ***Journal of Computer and System Sciences***, Vol. 13, pp. 300-317, 1976.

[4] Rivest, R., Shamir, A. and Adleman, L., "A method for obtaining digital signature and public key cryptosystems," ***Communications of the ACM***, Vol. 21(2), pp. 120-126, 1978.

[5] Shamir, A. and Tauman, Y., "Improved online / offline signature schemes," ***Advances in Cryptology-CRYPTO'01***, LNCS 2139, pp. 355-367, 2001.

[6] Yang, F. Y., "Improvement on a trapdoor hash function," ***International Journal of Network Security***, Vol. 9, No. 1, July, pp. 17-21, 2009.

[7] Yearly Report on Algorithms and Keysizes (2009), D.SPA.7 Rev 1.0, ICT-2007-216676 ECRYPT, 07/2009.