使用三个数域的数域筛算法*

顾海华[1,2],谷大武[1],谢文录[2],李升[1],严家驹[1]
(1. 上海交通大学 计算机科学与工程系,上海 200240;
2. 上海华虹集成电路有限责任公司,上海 201203)

摘 要:大整数分解难题是RSA密码的数学安全基础。目前数域筛算法是分解365比特以上大整数的最有效方法,然而它的时间复杂度仍然是亚指数的。对于目前普遍使用的1024比特以上大整数,数域筛算法还不能分解,所以研究数域筛算法具有重要的意义。现有的一般数域筛算法普遍使用两个数域,对多个数域的研究极少。一般数域筛算法经过修改可以使用三个数域,即两个代数数域和一个有理数域。分析表明:修改后的数域筛算法与原来的一般数域筛算法在时间复杂度上处于同一量级。但修改后的数域筛算法有更多地方可以合并计算,所以计算速度更快了。通过两个实验也验证了这一结论。

关键词:公钥密码;RSA密码;整数分解;数域筛算法
中图分类号:TP 309.7 **文献标志码**:A **文章编号**:1001-2486(2012)02-0001-05

The number field sieve with three number fields

GU Haihua[1,2], *GU Dawu*[1], *XIE Wenlu*[2], *LI Sheng*[1], *YAN Jiaju*[1]
(1. Department of Computer Science & Engineering, Shanghai Jiaotong University, Shanghai 200240, China;
2. Shanghai Huahong Integrated Circuit Co., Ltd., Shanghai 201203, China)

Abstract: The mathematical security of the RSA cryptosystem is based on the problem of factoring large integers. At present, the number field sieve is the most efficient algorithm known for factoring integers larger than 365 bits, while its time complexity is still sub-exponential. Integers larger than 1024 bits, which are widely used in RSA cryptosystem, cannot be factored by means of the number field sieve. So it is significant to study the number field sieve. Now, the general number field sieve often uses two number fields, while multiple number fields are seldom considered. The general number field sieve, through adaptation, can use three number fields, i.e. two algebraic number fields and one rational number field. Analysis shows that the time complexity of the modified number field sieve and the general number field sieve are at the same level. However, the modified number field sieve can combine more computation, so it can save more time in practice. Finally, the result is verified by two experiments.

Key words: public key cryptography; RSA cryptosystem; integer factorization; number field sieve algorithm

RSA密码是由三位图灵奖获得者Rivest, Shamir和Adleman于1978年提出[1]的,目前已经广泛应用于网上银行、金融IC卡、移动支付、电子护照等高安全领域。RSA密码的数学安全性基于大整数分解问题,这里的大整数是两个素因子的乘积。随着整数的增大,这个问题的困难性急剧增加。为了鼓励人们更深入地研究大整数分解问题,RSA实验室在1991年向全世界发布悬赏,任何人或组织,无论使用多少计算机,只要最先分解悬赏列表中的大整数,就可以得到相应的奖金。悬赏列表中包含54个大整数,这些整数的范围从330比特到2048比特。悬赏发布的同一年,Lenstra带领他的研究小组使用二次筛法就分解了330比特的整数;随后,人们分解的大整数不断增大,397,463,512等比特的RSA整数相继被分解[2-4];到2009年底已经能够分解768比特RSA整数[5]。分解整数的算法有很多,譬如试除法、费马分解法、$p-1$算法、连分数分解法[6]、椭圆曲线分解法[7]、二次筛法[8]、数域筛算法[9]等。其中数域筛算法是目前分解365比特以上大整数的最有效方法。

1988年,Pollard提出数域筛思想。接着,他与Lenstra, Lenstra, Manasse合作,于1990年在STOC会议上正式提出数域筛算法[9]。当时的数域筛只能分解一类特殊的整数,这类整数的形式为$n=b^c\pm1$,其中b是一个小的整数。后来,他们

* **收稿日期**:2011-09-14
基金项目:教育部博士点专项基金项目(200802480019);
作者简介:顾海华(1981—),男,上海人,博士,E-mail: guhaihua@shhic.com
谷大武(通信作者),男,教授,博士,博士生导师,E-mail:dwgu@sjtu.edu.cn

四人利用这个特殊数域筛算法成功分解了第9个费马数 $F_9 = 2^{2^9} + 1$，它的长度是513比特[10]。1993年，Buhler，Lenstra，Pomerance把特殊数域筛进行了推广[11]，使得它可以分解一般的整数。分解768比特整数的方法就是一般数域筛，它的时间复杂度是亚指数，具体为 $\exp\left(\left(\left(\frac{64}{9}\right)^{\frac{1}{3}} + o(1)\right)(\log_2 n)^{\frac{1}{3}}(\log_2 \log_2 n)^{\frac{2}{3}}\right)$。起初，一般数域筛使用一个有理数域和一个代数数域；后来，Montgomery，Willianms等研究了如何使用两个代数数域[12-13]，文献[14]提出了使用多个数域。

本文对文献[14]的构造法做了修改，采用两个代数数域和一个有理数域，通过复杂度分析发现，修改后的时间复杂度是 $\exp\left(\left(\left(\frac{64}{9}\right)^{\frac{1}{3}} + o(1)\right)(\log_2 n)^{\frac{1}{3}}(\log_2 \log_2 n)^{\frac{2}{3}}\right)$，虽然与一般数域筛的时间复杂度在同一量级，但由于修改后的数域筛合并了更多的相同计算，从而 $o(1)$ 更小了。由于 $o(1)$ 是在指数上，微小的减小都会明显提高计算效率。

1　预备知识

设 $f(x) = a_n x^n + \cdots + a_1 x + a_0$ 是一个有理系数多项式，如果 α 是 $f(x) = 0$ 在复数域上的一个根，则 α 是一个代数数。代数数域 $Q(\alpha)$ 是有理数域的 n 次扩张。现在取 $f(x)$ 是一个不可约整系数多项式，α 是 $f(x) = 0$ 在复数域上的一个根。如果整数 m 满足 $f(m) \equiv 0 \bmod n$，则存在一个自然环同态 $\varphi: Z[\alpha] \to Z/nZ; \varphi(\alpha) \equiv m \bmod n$。

数域筛的思想是寻找一个非空集合 S，包含两两互素的整数对 (a,b) 满足 $\prod_{(a,b)\in S}(a + bm)$ 在 Z 中是一个平方数，$\prod_{(a,b)\in S}(a + b\alpha)$ 在 $Z[\alpha]$ 中是一个平方数。

如果令 $X^2 = \prod_{(a,b)\in S}(a + bm)$，$\beta^2 = \prod_{(a,b)\in S}(a + b\alpha)$，于是 $\varphi(\beta^2) = \varphi(\prod_{(a,b)\in S}(a + b\alpha)) = \prod_{(a,b)\in S}(a + b\varphi(\alpha)) \equiv \prod_{(a,b)\in S}(a + bm) \bmod n \equiv X^2 \bmod n$。

记 $Y = \varphi(\beta) \bmod n$，则 $Y^2 \equiv X^2 \bmod n$，于是可以通过 $\gcd(X - Y, n)$ 求 n 的因子。

为了清晰、简洁地叙述数域筛算法，还需要定义如下概念：

对于代数数域 $Q(\alpha)$ 中的元素 β，它的范数 $N(\beta)$ 定义为 $N(\beta) = \sigma_1(\beta)\sigma_2(\beta)\cdots\sigma_d(\beta)$，其中 σ_i 是 $Q(\beta)$ 到复数域 C 的嵌入，这里嵌入是指保持代数结构不变的单射。

如果整数 n 的素因子都小于 y，则称 n 为 y-光滑的整数。定义集合 $S = \{$数对$(p,r): p \in [1, y], r \in \{0, \cdots, p-1\}$ 且 $f(r) \equiv 0 \bmod p\}$。令 $a, b \in Z$，且 $\gcd(a,b) = 1$。设 $(p,r) \in S$，定义 $e_{p,r}(a + b\alpha) = \begin{cases} ord_p(N(a + b\alpha)) & \text{当 } a + br \equiv 0 \bmod p \\ 0 & \text{其他} \end{cases}$

其中 $ord_p(k)$ 表示整数 k 的素因子分解式中 p 的个数。由 $e_{p,r}$ 的定义知：

$$N(a + b\alpha) = \pm \prod_{p,r} p^{e_{p,r}(a+b\alpha)}$$

2　改进的数域筛算法

在使用数域筛分解整数的过程中，我们发现集合 S 中 (a,b) 对往往不够多，从而导致构造不出满足条件的平方数。为此，我们对数域筛算法进行了修改。我们的思想是：

使用三个不可约多项式 $f_i(x)$，$i = 1,2,3$，而 α_1, α_2 分别是 $f_1(x)$，$f_2(x)$ 在复数域上的根，相应地可以构造两个代数数域；$f_3(x)$ 是一个线性多项式，即 $px - m$，这里 p 是一个正整数。如果 $m \cdot p^{-1} \bmod n$ 满足 $f_i(m \cdot p^{-1}) \equiv 0 \bmod n$，则存在自然环同态 $\varphi_i: Z[\alpha_i] \to Z/nZ; \varphi_i(\alpha_i) \equiv m \cdot p^{-1} \bmod n$，$i = 1,2$。

寻找两个非空集合 S_1, S_2，包含两两互素的整数对 (a,b) 满足 $\prod_{(a,b)\in S_1}(a + bm \cdot p^{-1} \bmod n)$ $\prod_{(a,b)\in S_2}(a + bm \cdot p^{-1} \bmod n)$ 在 Z 中是一个平方数；$\prod_{(a,b)\in S_1}(a + b\alpha_1)$ 在 $Z[\alpha_1]$ 中是一个平方数；$\prod_{(a,b)\in S_2}(a + b\alpha_2)$ 在 $Z[\alpha_2]$ 中是一个平方数。若令 $X^2 = \prod_{(a,b)\in S_1}(a + bm \cdot p^{-1} \bmod n)\prod_{(a,b)\in S_2}(a + bm \cdot p^{-1} \bmod n)$ 以及 $\beta_1^2 = \prod_{(a,b)\in S_1}(a + b\alpha_1)$，$\beta_2^2 = \prod_{(a,b)\in S_2}(a + b\alpha_2)$，于是 $\varphi_1(\beta_1^2) = \varphi_1(\prod_{(a,b)\in S_1}(a + b\alpha_1)) = \prod_{(a,b)\in S_1}(a + b\varphi_1(\alpha_1)) \equiv \prod_{(a,b)\in S_2}(a + bm \cdot p^{-1}) \bmod n$；$\varphi_2(\beta_2^2) = \varphi_2(\prod_{(a,b)\in S_2}(a + b\alpha_2)) = \prod_{(a,b)\in S_2}(a + b\varphi_2(\alpha_2)) \equiv \prod_{(a,b)\in S_2}(a + bm \cdot p^{-1}) \bmod n$。

记 $Y_i = \varphi_i(\beta_i) \bmod n$，$i = 1,2$，则 $Y_1^2 Y_2^2 \equiv X^2 \bmod n$，于是可以通过 $\gcd(X - Y_1 Y_2, n)$ 求 n 的因子。以下是算法的详细过程：

算法1　改进的数域筛算法

输入: 正合数 n, 以及参数 u,y,z;

输出: n 的一个非平凡因子或返回“分解失败”。

步骤 1:这一步将产生三个整系数多项式 $f_i(x),i=1,2,3$,其中 $f_1(x)=a_{1d}x^d+\cdots+a_{10}$, $f_2(x)=a_{2d}x^d+\cdots+a_{20}$的次数 $d>1$,而 $f_3(x)$是一个线性多项式,即 $px-m$。这里 p 是一个正整数, $m\approx\left(\frac{2n}{a_{1d}+a_{2d}}\right)^{\frac{2}{d}}$。而且 $f_i(x) \bmod n$ 有一个共同的根 $m\cdot p^{-1} \bmod n$。生成多项式 f_1,f_3 的方法可以参考文献[15];而 f_2 则是 f_1 和 f_3 的一个线性组合。不妨设 $f_i(x)$ 在 $Z[x]$ 中不可约,否则如果存在 $g(x)$ 整除某个 $f_i(x)$,则 $g(m\cdot p^{-1}\bmod n)$就是 n 的一个因子,那么分解 n 就完成了。

步骤 2:选取因子基。选择整数 y,z, 并且使得以下集合 QCB 的元素个数为 $\left[3\log_2\frac{2}{\log_2 n}\right]$。定义有理因子基:RFB = {素数 p: $p\in[1,y]$};代数因子基:AFB = {数对(p,r):素数 $p\in[1,y]$, $r\in\{0,\cdots,p-1\}$且$f(r)\equiv 0 \bmod p$};二次特征基:QCB = {数对(q,s):素数 $q\in[y,z]$, $s\in\{0,\cdots,q-1\}$ 且$f(s)\equiv 0 \bmod q$}。

步骤 3:寻找整数对(a,b)满足 $\gcd(a,b)=1$ 且$|a|\leqslant u$,$0<b\leqslant u/2$, 使得

$$|(pa-bm)N(a+b\alpha_1)|=\left|(pa-bm)b^d f_1\left(\frac{a}{b}\right)\right| \tag{1}$$

或

$$|(pa-bm)N(a+b\alpha_2)|=\left|(pa-bm)b^d f_2\left(\frac{a}{b}\right)\right| \tag{2}$$

是 y-光滑的。把式(1)确定的(a,b)放入集合 T_1,式(2)确定的(a,b)放入集合 T_2。

步骤 4:对于每一个数对 $(a,b)\in T_i$,按照以下方法构建矩阵:

每一个数对(a,b)对应一行,而第一列由 $pa-bm$ 的正负号确定,即如果 $pa-pm>0$,则第一列为 0,否则为 1;接下来#RFB 列由 $ord_p(pa-bm) \bmod 2$ 确定,$p\in$ RFB;随后#AFB 列由 $e_{p,r}(a+b\alpha_i) \bmod 2$ 确定,$(p,r)\in$ AFB;最后#QCB 列由 Legendre 符号$\left(\frac{a+bs}{q}\right)$确定,$(q,s)\in$ QCB。对于 T_1,T_2 中不同集合的(a,b)对,AFB 相应的列不能共用,其余的列可以共享。这样构造的矩阵有 1 +#RFB +2 ×#AFB +#QCB 列。矩阵构建之后,在行向量中找出一组非平凡的线性相关向量,并把这组向量所对应的(a,b)记为集合 S_i,显然 $S_i\subseteq T_i$,$i=1,2$;如果找不到,则返回“分解失败”。

步骤 5:记 $Y_i=f_i'(\alpha_i)\prod_{(a,b)\in S_i}(a+b\alpha_i)$,求 Y_i 的平方根$\beta_i=\sum_{j=0}^{d-1}b_{ij}\alpha_i^j$,$i=1,2$。如果求不出,则返回“分解失败”。

步骤 6:由等式 $c^2=\prod_{i=1}^{2}[f'_i(m\cdot p^{-1} \bmod n)\prod_{(a,b)\in S_i}(a+bm\cdot p^{-1}) \bmod n]$ 求出 $c \bmod n$。

步骤 7:计算最大公因子 $\gcd[c-\sum_{j=0}^{d-1}b_{1j}(m\cdot p^{-1}\bmod n)^j\sum_{j=0}^{d-1}b_{2j}(m\cdot p^{-1}\bmod n)^j,n]$, 如果它是整数 n 的非平凡因子,则分解成功;否则从集合 S_i 中选出一个元素,然后从集合 T_i 中删除它,跳转到步骤 4。

3　复杂度分析

对于实数 $\gamma,v\in R$ 且 $0\leqslant v\leqslant 1$, 当 $n\to\infty$ 时,函数 $L_n[v,\gamma]$ 定义为 $\exp((\gamma+o(1))(\log_2 n)^v(\log_2\log_2 n)^{1-v})$。设 $x\geqslant 1,y\geqslant 1$,令 $\psi(x,y)$ 表示集合$\{n\mid n\leqslant x$ 且 n 是 y-光滑的整数$\}$中元素的个数。

引理 1[11]　设 g 是一个关于 $y\geqslant 2$ 的函数,且满足 $g(y)\geqslant 1$,当 $y\to\infty$ 时,$g(y)=y^{1+o(1)}$,那么当 $x\to\infty$ 时,对于所有的 $y\geqslant 2$,有 $\frac{xg(y)}{\psi(x,y)}\geqslant L_x\left[\frac{1}{2},\sqrt{2}+o(1)\right]$。

引理 2[11]　令实数 l,k,v 满足 $l\geqslant 1$,$\log_2 k\geqslant 1$,$\log_2 v\geqslant 1$,如果$\frac{v^2}{\log_2 v}=kv+1$,则当 $k+1\to\infty$ 时,$2v=(1+o(1))(k\log_2 k+\sqrt{(k\log_2 k)^2+2l\log_2 l})$。

引理 3[11]　给定正整数 n,d 满足 $n>d^{2d^2}>1$,令 $u=u(n,d)>2$ 且 $y=y(n,d)\geqslant 2$,$x=x(n,d)=2dn^{2/d}u^{d+1}$,若当 $y\to\infty$ 时,对于函数 $g(y)=y^{1+o(1)}\geqslant 1$ 满足$\frac{u^2\psi(x,y)}{x}\geqslant g(y)$,则 $n\to\infty$ 时$2\log_2 u\geqslant(1+o(1))(d\log_2 d+\sqrt{\Delta})$,其中 $\Delta=(d\log_2 n^{\frac{1}{d}})^2+4\log_2(n^{\frac{1}{d}})\log_2\log_2(n^{\frac{1}{d}})$。

假设满足 y-光滑的整数$(pa-bm)N(a+b\alpha)$在区间$[1,x]$内随机分布。由于此假设至今没人证明,因此以下的结论是一个猜想。

猜想:给定正合数 n,存在参数 u,y,z 使得算法 1 以一定的概率分解 n,且分解时间至多为

$L_n\left[\frac{1}{3},\left(\frac{64}{9}\right)^{\frac{1}{3}}+o(1)\right]$。

由算法1步骤3，当$|a|\leqslant u,0<b\leqslant\frac{u}{2}$，$|(pa-bm)N(a+b\alpha_i)|=\left|(pa-bm)b^d f_1\left(\frac{a}{b}\right)\right|\leqslant du^{d+1}a_{id}\left(\frac{2n}{a_{1d}+a_{2d}}\right)^{\frac{1}{d}}\leqslant du^{d+1}a_{id}n^{1/d}$。

又根据文献[15]中算法3.6知：$a_{id}\leqslant\left(\frac{M^{2d-2}}{n}\right)^{\frac{1}{d-3}}\leqslant n^{1/(d+1)}\leqslant n^{1/d}$，这里$M<n^{1/(d+1)}$。所以

$$|(pa-bm)N(a+b\alpha_i)|\leqslant du^{d+1}n^{2/d}$$

取$x=du^{d+1}n^{2/d}$。现在假设满足y-光滑的整数$(pa-bm)N(a+b\alpha_i)$在区间$[1,x]$内随机分布。根据$\psi(x,y)$的定义可知，给定一个小于等于x的正整数，它满足y-光滑的概率为$\psi(x,y)/x$，而我们所搜索满足$|a|\leqslant u,0<b\leqslant u/2,\gcd(a,b)=1$的$(a,b)$对有$cu^2/2$个，这里$c=12/\pi^2$。所以我们可以认为在步骤3中集合$T_1\cup T_2$的大小为$2cu^2\psi(x,y)/x$，这就意味着步骤4中矩阵的行数为$\left(\frac{u^2\psi(x,y)}{x}\right)^{1+o(1)}$。

另一方面，根据引理3，我们取$u=y=\exp\left(\left(\frac{1}{2}+o(1)\right)(d\log_2 d+\sqrt{\Delta})\right)$。由步骤4中矩阵的构造方法可知，矩阵的列数为$1+\#\mathrm{RFB}+2\times\#\mathrm{AFB}+\#\mathrm{QCB}$，而$\#\mathrm{RFB}=\pi(y)\leqslant y$，$\#\mathrm{AFB}\leqslant dy$，$\#\mathrm{QCB}\leqslant(3\log_2 n)/\log_2 2$，$d<\log_2 n=y^{o(1)}$。于是$1+\#\mathrm{RFB}+2\times\#\mathrm{AFB}+\#\mathrm{QCB}\leqslant y^{1+o(1)}\leqslant\left(\frac{u^2\varphi(x,y)}{x}\right)^{1+o(1)}$，即矩阵的行数大于等于矩阵的列数，从而对应的方程组有解，也就意味着大整数n在参数u,y,z下可以分解。

最后，我们分析分解所需要时间。在算法1中，步骤3占用的时间最多，由于搜索区间为$|a|\leqslant u,0<b\leqslant u/2$，所以其时间复杂度为$u^{2+o(1)}$；又因为相对于步骤3来说，其余步骤的耗时相对较少，所以算法1的时间复杂度为$\exp((1+o(1))(d\log_2 d+\sqrt{\Delta}))$。当$n$充分大时，可以取$d=\left(\frac{(3+o(1))\log_2 n}{\log_2\log_2 n}\right)^{\frac{1}{3}}$，于是算法1的时间复杂度

$$\begin{aligned}&\exp((1+o(1))(d\log_2 d+\sqrt{\Delta}))\\&\leqslant\exp((1+o(1))(\sqrt{5}+1)d\log_2 d)\\&\leqslant\exp((1+o(1))4d\log_2 d)\\&\leqslant L_n\left[\frac{1}{3},\left(\frac{64}{9}\right)^{\frac{1}{3}}+o(1)\right]\end{aligned}$$

从以上叙述过程可以看出，虽然算法1的时间复杂度与一般数域筛算法相同，但其中无穷小量$o(1)$是不完全一样的。在步骤3中，我们把原来一般数域筛的条件：$|(pa-bm)N(a+b\alpha)|=\left|(pa-bm)b^d f\left(\frac{a}{b}\right)\right|$满足$y$-光滑，放宽为：$|(pa-bm)N(a+b\alpha_1)|=\left|(pa-bm)b^d f_1\left(\frac{a}{b}\right)\right|$或$|(pa-bm)N(a+b\alpha_2)|=\left|(pa-bm)b^d f_2\left(\frac{a}{b}\right)\right|$是$y$-光滑，从而增大了满足$y$-光滑的概率，可以获得更多的$(a,b)$对。因此，如果一般数域筛的搜索区间为$|a|\leqslant u,0<b\leqslant u$，那么算法1的搜索区间可以缩小为$|a|\leqslant u,0<b\leqslant u/2$；而且$pa-bm$在条件$|(pa-bm)N(a+b\alpha_1)|$和$|(pa-bm)N(a+b\alpha_2)|$中都出现，算法实现过程中可以合并相同的计算过程，从而减少计算时间，提高运算效率。

4　实验

为了验证对一般数域筛的改进是否有效，我们做了以下实验。实验所用的大整数取自RSA公司于1991年公布的悬赏列表。我们选用了其中的RSA-100，即整数的大小是100位，332比特：

RSA-100 = 1522605027922533360535618378132637429718068114961380688657908494580122963258952897654000350692006139

生成多项式：

$f_1=18000x^5-818393050x^4-5150812346923x^3+34865932828331120x^2+458219032633868074 38x+5768017974969449823990 0$

$f_3=8483526557x-9670889371271295019$

取$f_2=f_1+5f_3$；因子基的上界定为$y=2^{20}$，取$u=100$作为计算样本区间的上界。假设一般数域筛选用了$f_1(x),f_3(x)$，当搜索区间为$|a|\leqslant u,0<b\leqslant u$时，找到的$(a,b)$对个数为130；而我们修改后的数域筛只搜索了$|a|\leqslant u,0<b\leqslant u/2$，却找到了144个$(a,b)$对，并且搜索耗时更少。为了避免因多项式不同而产生的误差，我们又假设一般数域筛使用$f_2(x),f_3(x)$，当搜索区间为$|a|\leqslant u,0<b\leqslant u$时，找到的$(a,b)$对个数为116，同样少于144个。实验数据总结在表1中。

为了避免因大整数不同而产生误差，我们又选用了RSA-768进行实验，它是一个232位，即

768 比特的整数,它也是 RSA 公司悬赏列表中的整数之一。

表 1　RSA－100 实验数据

Tab. 1　Experimental data of RSA－100

算法	耗 时 (s)	(a,b)对 (个)	速度 (个/s)
一般数域筛 f_1, f_3	114	130	1.14
一般数域筛 f_2, f_3	127	116	0.91
修改数域筛 f_1, f_2, f_3	76	144	1.89

RSA－768 = 123018668453011775513049495838496272077285356959533479219732245215172640005072636575187452021997864693899564749427740638459251925573263034537315482685079170261221429134616704292143116022212404792747377940806653514195974598569021434131888

多项式引用自文献[5]:

$f_1 = 265482057982680x^6 + 12765093607683211888x^5 - 5006815697800138351796828x^4 - 46477854471727854271772677450x^3 + 6525437261935989397109667371894785x^2 - 1818577935208859435672601886243480305 4x - 27756526679154388199521619971380110334 3120$

$f_3 = 34661003550492501851445829x - 12911874565800212231635477915748108 81$

令 $f_2 = f_1 - 188264522xf_3$;取 $y = 2^{40}$, $u = 200$。一般数域筛使用的搜索区间为 $|a| \leqslant u, 0 < b \leqslant u$;而我们修改后的数域筛使用的搜索区间为 $|a| \leqslant u, 0 < b \leqslant u/2$。修改后的数域筛以更少的时间找到了更多的$(a,b)$对,见表 2。

表 2　RSA－768 实验数据

Tab. 2　Experimental data of RSA－678

算法	耗　时 (s)	(a,b)对 (个)	速　度 (个/s)
一般数域筛 f_1, f_3	5149	18	0.0035
一般数域筛 f_2, f_3	6043	8	0.0013
修改数域筛 f_1, f_2, f_3	3106	21	0.0068

从表 2 中数据来看,修改后的数域筛在寻找(a,b)对这一步可以减少约 48% 的运算时间。

5　结　论

本文对一般数域筛进行了修改,分析和实验都表明:修改后的数域筛在主体筛选部分比一般数域筛具有更高的效率。当然,修改后,矩阵会增大,如何提高这一部分的计算效率也是一个有意义的课题。

参考文献(References)

[1] Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public key cryptosystems [J]. Communications of the ACM, 1978, 21 (2): 120－126.

[2] Denny T, Dodson B, Lenstra A K, et al. On the factorization of RSA-120 [C]// Proc of Crypto 93, LNCS 773, Springer-Verlag, 1993:166－74.

[3] Cavallar S, Dodson B, Lenstra A, et al. Factorization of RSA-140 using the number field sieve [C]// Proc of Asiacrypt'99, LNCS 1716, Springer-Verlag, 1999: 195－207.

[4] Cavallar S, Dodson B, Lenstra A, et al. Factorization of a 512-bit RSA modulus [C]// Proc of Eurocrypt 2000, LNCS 1807, Springer-Verlag, 2000: 1－17.

[5] Kleinjung T, Aoki K, Franke J, et al. Factorization of a 768-bit RSA Modulus [C]// Proc of Crypto 2010, LNCS 6223, Springer-Verlag, 2010: 333－350.

[6] Michael M A, Brillhart J. A method of factoring and the factorization of F7[J]. Math. Comp. ,1975,29(129):183－205.

[7] Lenstra H W. Factoring integers with elliptic curves [J]. Ann. Math., 1987, 126 (3): 649－673.

[8] Pomerance C. Analysis and comparison of some integer factoring algorithms [R]. Computational Methods in Number Theory, Part I, Math. Centre Tract 154, Amsterdam, 1982: 89－139.

[9] Lenstra A K, Lenstra H W, Manasse M S, et al. The number field sieve [C]// Proc of STOC, 1990: 564－572.

[10] Lenstra A K, Lenstra H W, Manasse M S, et al. The factorization of the ninth fermat number [J]. Math. Comp., 1993, 61: 319－349.

[11] Buhler J P, Lenstra H W, Pomerance C. Factoring integers with the number field sieve [C]// The development of the number field sieve, LNCS 1554, Springer-Verlag, 1993 : 50－94.

[12] Murphy B A, Brent R P. On quadratic polynomials for the number field sieve [J]. Computing Theory ACSC, 1998, 20 (3): 199－213.

[13] Williams R S. Cubic polynomials in the number field sieve [D]. lubbock: Texas Tech university,2010.

[14] Elkenbracht-Huizing R M. A multiple polynomial general number field sieve [C]// Proc of ANTS-II, LNCS 1122, Springer-Verlag, 1996: 99－114.

[15] Kleinjung T. On polynomial selection for the general number field sieve [J]. Math. Comp., 2006, 75: 2037－2047.