Experience in Factoring Large Integers Using Quadratic Sieve

D. J. Guan
Department of Computer Science,
National Sun Yat-Sen University,
Kaohsiung, Taiwan 80424
guan@cse.nsysu.edu.tw

April 19, 2005

Abstract

GQS is a set of computer programs for factoring "large" integers. It is based on multiple polynomial quadratic sieve. The current version, 3.0, can factor a 82-decimal-digit integer in a PC with AMD 1.8G Hz processor and 512 MB main memory in one day. The largest number I have factored using GQS is RSA-130, a 130-digit integer. This was done in three PC clusters, two of which has 16 nodes and the other has 64 nodes. In this talk I will describe how the GQS is implemented and my experience in using GQS.

Theory of Quadratic Sieve

Quadratic sieve is an "efficient" algorithm for factoring integers up to about 120 decimal digits.

Let n be the integer to be factored. The algorithm first finds a pair of congruent squares

$$x^2 \equiv y^2 \pmod{n}.$$

Suppose that

$$x \not\equiv \pm y \pmod{n}$$

Then $\gcd(n, x-y)$ and $\gcd(n, x+y)$ are the non-trivial factor of n.

For example, $189^2 \equiv 50^2 \pmod{33221}$, $\gcd(33221, 189 - 50) = 139$ and $\gcd(33221, 189 + 50) = 239$ are two factors of 332221.

Theory of Quadratic Sieve

Assume that n is a product of two different primes.

$x^2 \equiv y^2 \pmod{n} \Rightarrow x^2 - y^2 \equiv 0 \pmod{n} \Rightarrow n \mid (x-y)(x+y)$

$x \not\equiv \pm y \pmod{n} \Rightarrow n \nmid (x-y) \quad \text{and} \quad n \nmid (x+y)$

Thus, $x-y$ and $x+y$ each contains only one factor of n, but not both.

Therefore, $\gcd(n, x-y)$ and $\gcd(n, x+y)$ are non-trivial factors of n.

Find Congruent Squares

First, find a set of integers whose square modulo n are b-*smooth*:

$$S = \{x \mid \text{every prime factor of } x^2 \bmod n \text{ is bounded by } b\}.$$

Let $P = \{p \mid p \text{ is prime and } p \leq b\}$.

We then compute a set of S such that the residue $x_i^2 \bmod n$ of each $x_i \in S$ can be factored by using the primes in P.

$$x_i^2 \bmod n = \prod_{j=1}^{s} p_j^{e_{i,j}}.$$

The next step is to find a subset T of S such that

$$\prod_{x_i \in T} x_i^2 \bmod n = \prod_{x_i \in T} \left(\prod_{j=1}^{s} p_j^{e_{i,j}} \right) = \prod_{j=1}^{s} p_j^{c_j},$$

in which each $c_j = \sum_{x_i \in T} e_{i,j}$ is even.

If $|S| > |P|$, then there exists a subset $T \subset S$ satisfying the the above conditions.

After T is computed, we can then find two congruent squares

$$\left(\prod_{x_i \in T} x_i \right)^2 \equiv \left(\prod_{j=1}^{s} p_j^{c_j/2} \right)^2 \pmod{n}.$$

Find b-Smooth Integers — Simple Method

for $i = 1, 2, \ldots$ **do**
  $x_i = \lfloor \sqrt{n} \rfloor + i$
  $r = x_i^2 \bmod n$
  for $j = 1, 2, \ldots, s$ **do**
    $e_{i,j} = 0$
    while $(p_j \mid r)$ $\{e_{i,j} = e_{i,j} + 1;\ r = r/p_j;\}$
  if $(r = 1)$ **then** $x_i^2 \bmod n = \prod_{j=1}^{s} p_j^{e_{i,j}}$

An Example

Let $n = 33221$ and $B = \{2, 3, 5, 7\}$. Consider $x_i = \left[\sqrt{33221}\right] + i = 182 + i$. The followings is a list of x_i's whose square modulo n can be factoed by primes in B.

$$
\begin{aligned}
(189)^2 &\equiv 2500 &&\equiv (2)^2(5)^4 \\
(378)^2 &\equiv 10000 &&\equiv (2)^4(5)^4 \\
(409)^2 &\equiv 1176 &&\equiv (2)^3(3)^1(7)^2 \\
(567)^2 &\equiv 22500 &&\equiv (2)^2(3)^2(5)^4 \\
(682)^2 &\equiv 30 &&\equiv (2)^1(3)^1(5)^1 \\
(802)^2 &\equiv 12005 &&\equiv (5)^1(7)^4 \\
(818)^2 &\equiv 4704 &&\equiv (2)^5(3)^1(7)^2 \\
(835)^2 &\equiv 32805 &&\equiv (3)^8(5)^1 \\
(845)^2 &\equiv 16384 &&\equiv (2)^{14} \\
(983)^2 &\equiv 2880 &&\equiv (2)^6(3)^2(5)^1 \\
(1169)^2 &\equiv 4500 &&\equiv (2)^2(3)^2(5)^3 \\
&\vdots
\end{aligned}
$$

An Example: Factoring

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

$189^2 \equiv 2^2 5^4 \equiv 50^2$
$189^2 - 50^2 \equiv 0 \bmod 33221$
$(189 - 50)(189 + 50) \equiv 0 \bmod 33221$
$(139)(239) = k(33221)$
$33221 = 139 \times 239$

$$802^2 \equiv 5^1 7^4$$
$$835^2 \equiv 3^8 5^1$$

$$(802 \cdot 835)^2 \equiv (3^4 \cdot 5 \cdot 7^2)^2$$
$$(669670)^2 \equiv (19845)^2$$
$$(5250)^2 \equiv (19845)^2$$
$$(5250 - 19845)(5250 + 19845) = k \cdot 33221$$
$$(-14595)(20595) = k \cdot 33221$$
$$(-14595, 33221) = 139$$
$$(20595, 33221) = 239$$

$$409^2 \equiv 2^3 3^1 7^2$$
$$682^2 \equiv 2^1 3^1 5^1$$
$$835^2 \equiv 3^8 5^1$$

$$(409 \cdot 682 \cdot 835)^2 \equiv (2^2 \cdot 3^5 \cdot 5 \cdot 7)^2$$
$$(232913230)^2 \equiv (34020)^2$$
$$(799)^2 \equiv (799)^2$$

Implementation of GQS 3.0

1. The program computes a *multiplier* k, and try to factor kn, instead of n. The speed of the computation can be 3 times faster by selecting a proper value of k.
2. The factor basis P is chosen as the set of the first s primes p such that n is a quadratic residue modulo p.
3. The set S is computed by sieving with *multiple polynomials*.
4. The set T is computed by Gaussian elimination over $\mathbf{Z}/2\mathbf{Z}$.
5. In addition to the *fully factored* x_i's ($r = 1$), those x_i's with 1, 2, or even 3 large primes ($1 < r < p_m$), are also collected, where p_m is the upper bound for the primes.

Relations

We shall call each x_i together with their prime factors a *relation*, $x_i^2 \bmod n = \prod_{j=1}^{s} p_j^{e_{i,j}}$.

Both *fully factored* and *partially factored* relations are collected.

A partially factored relation contains some primes which are not in the factor basis P.

A relation can be transformed into a *normalized form* in which each $e_{i,j}$ is either 0 or 1.

This is done by multiplying $(p_j^{-\lfloor e_{i,j}/2 \rfloor})^2$ to both sides of the relation.

Six Programs in GQS

gqsinit determines the size s of the factor basis P, the value of the multiplier k, and the length m of the sieve interval.

It can also generate a number n, which is a product of two random primes, to be factored for testing the factoring programs.

gqssieve computes a set of numbers S whose square modulo kn is "p_s-smooth".

It is the program that runs most of the time.

The program *gqssieve* is designed to be able to continue its computation from the last time it was interrupted.

It can also be run in *parallel* in one machine with multiple CPU's or in different machines, such as a PC cluster.

gqsmerge combines all the relations computed by *gqssieve*. It also merges all the large primes in partially factored relations into the factor basis.

gqscycle eliminates useless relations.

A relation is *useless* if it has a prime p with odd power but no other relations have it.

gqsreduce reduces the size of the factor basis.

This is an effective way to reduce the size of the system of linear equations need to be solved in the factoring.

gqsfactor computes a subset of relations T, and then computes two congruent squares x and y, and finally compute the factors of n.

Implementation of gqssieve

GQS SIEVE 3.0 uses multiple pairs of polynomials $(f(x), g(x))$ for sieving.

Each pair of polynomials $f(x)$ and $g(x)$ must satisfy the following conditions.

1. $f(x)$ is a square,
2. $f(x) \neq g(x)$ but $f(x) \equiv g(x) \pmod{n}$.
3. The value of $g(x)$ should be small in the sieve interval $[-m, m]$.

In the simple method given above,

$$\begin{aligned} f(x) &= \left(x + \lfloor\sqrt{n}\rfloor\right)^2, \\ g(x) &= x^2 + 2\lfloor\sqrt{n}\rfloor + (\lfloor\sqrt{n}\rfloor)^2 - kn. \end{aligned}$$

The problem of using one pair of polynomials is that when x gets large, $g(x)$ will also be large, and large integers are unlikely to be smooth.

For example, it takes *one week* to factor a 43-digit integer in a PC.

Generation of the Polynomials

Let

$$\begin{aligned} h(x) &= (ax+b)^2 = a^2x^2 + 2abx + b^2, \\ g(x) &= ax^2 + 2bx + c, \\ a \cdot g(x) &= a^2x^2 + 2abx + ac. \end{aligned}$$

If $b^2 \equiv ac \pmod{n}$ then $h(x) \equiv a \cdot g(x) \pmod{n}$.

Assume that a is a square, $a \equiv d^2 \pmod{n}$.

Let $f(x) = (ax+b)^2(d^{-1})^2 = \left(ad^{-1}x + bd^{-1}\right)^2$.

Then $f(x) \equiv g(x) \pmod{n}$.

Generation of "good" Polynomials

1. minimize $\sup |g(x)|$ over $[-m, m]$, or
2. minimize $\int_{-m}^{m} |g(x)|\, dx$, or
3. minimize $\int_{-m}^{m} g^2(x)\, dx$,

subject to $b^2 - ac = kn$ (Note that $b^2 \equiv ac \pmod{n}$.)

The *optimal* solution is

$$a = \frac{\sqrt{2}}{2}\sqrt{kn}/m, \quad b = 0, \quad c = -\frac{\sqrt{2}}{4}m\sqrt{kn}.$$

Compute $d \approx \sqrt{\sqrt{kn/2}/m}$, such that d is prime, $\left(\frac{kn}{d}\right) = 1$.

Let $a = d^2$.

Find b such that $b^2 \equiv kn \pmod{a}$.

$b^2 - kn \equiv 0 \pmod{a} \Rightarrow b^2 - kn = ca \Rightarrow c = \frac{b^2 - kn}{a}$.

$g(x) = ax^2 + 2bx + c$

Since $b^2 - ac = kn$, $f(x) \equiv g(x) \pmod{n}$.

Compute b

Compute $h = (kn)^{(d+1)/4}$, that is, $h^2 = kn \pmod{d}$.

Compute $v = (2h)^{-1} \left(\frac{kn - h^2}{d} \right) \bmod d$.

Compute $b = h + vd \bmod d^2$.

$$b^2 \equiv h^2 + 2hvd + v^2 d^2 \equiv kn \pmod{d^2}$$

Theorem. Let p be a prime and $f(x)$ be a polynomial with integer coefficients. If $f(a) \equiv 0 \pmod{p^i}$ and $f'(a) \not\equiv 0 \pmod{p}$, then there exists a unique t so that $f(a + tp^i) \equiv 0 \pmod{p^{i+1}}$.

By the above theorem, we can "lift" a solution to a higher power by solving a linear congruent equation

$$f'(a)t \equiv -\frac{f(a)}{p^i} \pmod{p}.$$

Let r be a solution to $x^2 \equiv w \pmod{p^i}$, and $f(x) = x^2 - w$.

We need to solve a linear equation to lift the solution to $x^2 \equiv w \pmod{p^{i+1}}$.

$$2rt \equiv -\frac{r^2 - w}{p^i} \pmod{p}.$$

If $(2r, p) = 1$, then $t = -\frac{r^2 - w}{p^i}(2r)^{-1} \pmod{p}$.

Sieve

Problem: Find k such that $g(k)$ is smooth.

If $p^\alpha \mid g(x)$ then $p^\alpha \mid g(x + jp^\alpha)$ for any integer j.

In GQS, sieve is done by

1. initializing an array $w[-m..m]$ to 0, and
2. add $\left\lfloor \alpha \log(p_j) + 0.5 \right\rfloor$ to $w[x]$ iff $p_j^\alpha \mid g(x)$.

Ideally, we should do the sieve for all primes $p \in P$ for all α such that $p^\alpha < b$.

However, for $\alpha > 1$, GQS only do this for small set of p to reduce the computational time.

The following table shows the process of sieve.

	2	3	5	7	4	9
$g(-m)$	v					
$g(-m+1)$		v	v			v
$g(-m+2)$	v				v	
$g(-m+3)$				v		
$g(-m+4)$	v	v				
$g(-m+5)$						
$g(-m+6)$	v		v		v	
$g(-m+7)$		v				
$g(-m+8)$	v					
$g(-m+9)$						
$g(-m+10)$	v	v		v	v	v
$g(-m+11)$			v			
$g(-m+12)$	v					
$g(-m+13)$		v				
$g(-m+14)$	v				v	
$g(-m+15)$						
$\vdots$						
$g(m)$	v	v	v			

Trial Division

Trial division is applied to test the smoothness of each $g(k)$ when

$$w[k] > \left\lfloor \log \frac{m\sqrt{kn/2}}{p_s^t} \right\rfloor,$$

where t is the *threshold* value.

If $t \leq 2$ then most relations will be fully factored.

If $t > 2$, then relations may not be fully factored, but the partial factorization may also be useful.

Parallel Sieve

Sieve is very easy to be parallelized.

Let each process i do the sieve with the polynomial $g(x) = ax^2+bx+c$ in which a is within a given interval $[l_i, r_i]$.

Our experience shows that sieve is almost k times faster by using k computers.

Check Point and Restart

It usually take a long time to do the sieve when the number to be factored is large.

Checkpoint and restart are very easy to implement.

Save the value of a, and the program can start from there.

Implementation of *gqsfactor*

GQS FACTOR 3.0 use Gaussian elimination to find linear dependency after enough relations are found, and then factor n.

Each relation x_i in S is mapped to a vector v_i in $(\mathbf{Z}/2\mathbf{Z})^s$:

$$x_i = \prod_{j=1}^{s} p_j^{e_{i,j}} \quad \mapsto \quad v_i = (e_{i,1}, e_{i,2}, \ldots, e_{i,s}).$$

$n = 33221,\ B = \{2, 3, 5, 7\}$

$$
\begin{array}{lll}
(409)^2 & \equiv 1176 & \equiv (2)^3(3)^1(7)^2 \quad (1,1,0,0) \\
(567)^2 & \equiv 22500 & \equiv (2)^2(3)^2(5)^4 \quad (0,0,0,0) \\
(682)^2 & \equiv 30 & \equiv (2)^1(3)^1(5)^1 \quad (1,1,1,0) \\
(802)^2 & \equiv 12005 & \equiv (5)^1(7)^4 \quad (0,0,1,0) \\
(818)^2 & \equiv 4704 & \equiv (2)^5(3)^1(7)^2 \quad (1,1,0,0) \\
(835)^2 & \equiv 32805 & \equiv (3)^8(5)^1 \quad (0,0,1,0) \\
(845)^2 & \equiv 16384 & \equiv (2)^{14} \quad (0,0,0,0) \\
(983)^2 & \equiv 2880 & \equiv (2)^6(3)^2(5)^1 \quad (0,0,1,0) \\
(1169)^2 & \equiv 4500 & \equiv (2)^2(3)^2(5)^3 \quad (0,0,1,0) \\
(1223)^2 & \equiv 784 & \equiv (2)^4(7)^2 \quad (0,0,0,0) \\
(1227)^2 & \equiv 10584 & \equiv (2)^3(3)^3(7)^2 \quad (1,1,0,0) \\
(1327)^2 & \equiv 216 & \equiv (2)^3(3)^3 \quad (1,1,0,0)
\end{array}
$$

Assume that more than $|P| = s$ relations have been collected. These vectors cannot be linear independent.

Example of dependence:

$$\begin{array}{lll} (409)^2 & \equiv 1176 & \equiv (2)^3(3)^1(7)^2 \quad (1,1,0,0) \\ (682)^2 & \equiv 30 & \equiv (2)^1(3)^1(5)^1 \quad (1,1,1,0) \\ (835)^2 & \equiv 32805 & \equiv (3)^8(5)^1 \quad (0,0,1,0) \end{array}$$

Assume that there are t relations, $t > s$. A linear dependency can be found by solving the following equation for δ_i.

$$\delta_1 v_1 + \delta_2 v_2 + \cdots + \delta_t v_t = 0.$$

This is equivalent to solving a system of linear equation

$$A\delta = 0.$$

where

$$A = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_t \end{pmatrix}^T .$$

After Gaussian elimination, every column after s yields a solution.

We can also let

$$A = \left(\begin{array}{c|c} \begin{matrix} v_1 \\ v_2 \\ \vdots \\ v_t \end{matrix} & \mathbf{I} \end{array} \right),$$

where $\mathbf{I}$ is a $t \times t$ identity matrix.

After Gaussian elimination, any zero-row in the first s elements yields a solution.

Performance of GQS 3.0

I have factored an 82-digit integer in one day by a PC with AMD 1.8G Hz CPU and 512 MB memory.

I have also factored an 85-digit integer in 3 days in the same computer.

I have successfully factored a 110-digit integer. This was done in a PC cluster with about 50 available nodes. It took about a week to do the sieving. The merging of relations and the factoring took about a few hours.

Before the factoing of RSA-130, the largest number I have factored is an 115-digit integer. This is done in the same PC cluster in about three weeks.

Estimated Time for Factoring Integers Using GQS in a PC

$\log n$	50	60	70	80	90
time	0.01 h	0.16 h	2 h	22 h	210 h

$\log n$	100	110	120	130
time	80 d	600 d	4400 d	31000 d

Our experience shows that the estimation is quite close to the actual time required for factoring integers up to 115 digits.

Another Way to Generate the Polynomials

Let n be the integer to be factored, m be the length of the sieve interval. Let

$$\begin{aligned} a &\approx \sqrt{n/2}\,/m, \\ b^2 &\equiv n \pmod{a}, \quad 0 < b < \frac{a}{2}, \\ c &= \frac{b^2 - kn}{a}. \end{aligned}$$

Let

$$f(x) = (ax + b)^2, \tag{1}$$

$$g(x) = ax^2 + 2bx + c. \tag{2}$$

$$f(x) = a^2x^2 + 2abx + b^2, \tag{3}$$
$$a \cdot g(x) = a^2x^2 + 2abx + ac. \tag{4}$$

The difference of the above two equations, (4) $-$ (3), is

$$b^2 - ac = kn.$$

Therefore,

$$f(x) \equiv a \cdot g(x) \pmod{n}.$$

If a is a product of primes in the factor basis P, then we can sieve the polynomial g without inverting a.

1. Find a set of primes powers whose product is a "good" approximation to $\sqrt{n/2}/m$.
2. Solve $f(x) = ax^2 + 2bx + c \equiv 0 \pmod{p^\alpha}$ efficiently.

The advantage is that many values of b can be found for one value of a, and the next value of b can be computed by a small amount of work from the previous one.

Our experience shows that for factoring integers up to 90 digits, the new method is about 20

For integers larger than 100 digits, the method is not much faster then the old method.

This is due to the time for finding a good approximation to $\sqrt{n/2}\,/m$ become too large to be ignored.

The Challenges for Factoring Large Integers

For very large n, the factor basis P will be very large in order to find relations efficiently.

n	50	60	70	80	90
$\|P\|$	3000	4000	7000	15000	30000

n	100	110	120	130
$\|P\|$	51000	120000	245000	525000

In addition to sieving, the space and the time required to solve the system of linear equations will be the obstacles for factoring large integers.

Reduction of the Linear System

The technique used by GQS for solving large system of equations is to eliminate a subset of primes each time.

Let $Q \subseteq P$ be a subset of primes in the extended factor basis.

Do the Guassian elimination with respect to the primes in Q only.

This will not only reduce the space in the Guassian elimination, it will also reduce the computational time.

Our experience shows that when the matrix is still space, each round takes only 15 to 30 minutes. However, when the density of the matrix gets large, it takes a long time to finish.

$$
\begin{array}{lll}
(409)^2 & \equiv 1176 & \equiv (2)^3(3)^1(7)^2 \quad (1,1,0,0) \\
(567)^2 & \equiv 22500 & \equiv (2)^2(3)^2(5)^4 \quad (0,0,0,0) \\
(682)^2 & \equiv 30 & \equiv (2)^1(3)^1(5)^1 \quad (1,1,1,0) \\
(802)^2 & \equiv 12005 & \equiv (5)^1(7)^4 \quad (0,0,1,0) \\
(818)^2 & \equiv 4704 & \equiv (2)^5(3)^1(7)^2 \quad (1,1,0,0) \\
(835)^2 & \equiv 32805 & \equiv (3)^8(5)^1 \quad (0,0,1,0) \\
(845)^2 & \equiv 16384 & \equiv (2)^{14} \quad (0,0,0,0) \\
(983)^2 & \equiv 2880 & \equiv (2)^6(3)^2(5)^1 \quad (0,0,1,0) \\
(1169)^2 & \equiv 4500 & \equiv (2)^2(3)^2(5)^3 \quad (0,0,1,0) \\
(1223)^2 & \equiv 784 & \equiv (2)^4(7)^2 \quad (0,0,0,0) \\
(1227)^2 & \equiv 10584 & \equiv (2)^3(3)^3(7)^2 \quad (1,1,0,0) \\
(1327)^2 & \equiv 216 & \equiv (2)^3(3)^3 \quad (1,1,0,0) \\
(1364)^2 & \equiv 120 & \equiv (2)^3(3)^1(5)^1 \quad (1,1,1,0) \\
(1568)^2 & \equiv 270 & \equiv (2)^1(3)^3(5)^1 \quad (1,1,1,0) \\
(1589)^2 & \equiv 125 & \equiv (5)^3 \quad (0,0,1,0)
\end{array}
$$

In the factoring of the 115-digit integer, 192561 primes were used, and more than 1230000 relations had been collected. After elimination of useless relations, more then 480000 relations remain. The size of the final set of primes is about 450000.

The factorization is done by reducing 20000 primes at a time, until 80000 primes remain. Then gqsfactor was used to do the factoring.

The factoring of RSA-130

The final goal of our project is to factor the RSA-130, a 130-digit integer.

We do this in three main steps.

1. parameter selection
2. sieving
3. factoring

Parameter Selection

Parameter selection was done in October 2004.

We need to decide

1. p, the size of the factor basis,
2. l, the length of the sieve interval,

For each $p = 400000, 500000, 600000, 700000, 800000$, and for each $l = 20000000, 30000000, 40000000, 50000000, 60000000$, 768 polynomials are used to test the running time required and the relations obtained.

We try to find the values of p and l with largest *yield*,

$$s/(t \cdot p),$$

where s is the relations obtained, t is the time required to do the 768 polynomials.

The values of $s/(t \cdot p)$ are in the range $[0.92602 \times 10^{-6}, 4.34876 \times 10^{-6}]$.

Finally, we found that $p = 750000$ and $l = 30000000$ are the most efficient parameters, the value of $s/(t \cdot p) = 4.34876 \times 10^{-6}$.

Sieve

Sieve was done in three sets of PC clusters.

1. NCKU:
   (a) Model: PC made by a vender,
   (b) CPU: AMD XP, 1.4G Hz,
   (c) Memory: server 2G B, others: 1G B,
   (d) Disk: 20G B,
   (e) Number of nodes: 16 + server.
2. NCHC:
   (a) Model: IBM x335,

(b) CPU: 2 Intel Xeon, 2.8G Hz,

(c) Memory: 2G B,

(d) Disk: 60G B,

(e) Number of nodes: 150 + server, but only 16 nodes were dedicated to our project, each node has 2 CPU's.

3. NSYSU:

(a) Model: IBM A50 PC,

(b) CPU: Intel Pentium 4, 2.8G Hz,

(c) Memory: 256M B,

(d) Disk: 80 GB,

(e) Number of nodes: 64.

Performance of the Sieve

	NCKU	NCHC	NSYSU
Sieve started	2003/11/04	2004/02/02	2004/05/22
Sieve stopped	2004/07/27	2004/07/27	2004/07/25
Relations collected	1183652	1724298	2003123

Factoring

We started to collected relations and tried to find useful relations in June 2004.

On 2004/07/21, we found 1562743 useful relations with only 1556603 primes in these relations. The factoring process starts on 2004/07/21 at the National Super Computer Center.

On 2004/07/27, another factoring process started with 1738864 useful relations in which only 1659478 primes in these relations, and a new factoring process starts on 2004/08/12 at the National Sun Yat-Sen University's computer center.

First successful factoring ended on 2004/10/03 at the National Sun Yat-Sen university's computer center.

Partial elimination technique to gradually reduce the size of the factor basis. At each run 10000 primes are eliminated.

For the first 1200000 primes, each run takes about 15 to 30 minutes.

When the size of the factor basis was reduced to 400000, each run takes about 24 hours.

Parallel processing were used to reduce the size of the factor basis from 300000 down to 40000,

From 300000 down to 100000, each run takes about 1 to 2 day.

From 100000 down to 40000, each run takes about 6 hours in a 55 node PC cluster.

The final run takes less than 1.5 hours, and the factors of RSA-130 were found at the first solution set.

Acknowledgments

The implementation of GQS is based on GNU GMP and written in CWEB. GNU GMP performs very well in doing large integer arithmetic on most computers. CWEB is a very nice tool for both programming in C/C++ and documentation.

This project is supported in part by the National Science Council of the Republic of China.

References

1. Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1993.

2. Hans Riesel. *Prime Numbers and Computer Methods for Factorization*. Birkhäuser, 1994.

3. Robert D. Silverman. The multiple polynomial quadratic sieve. *Mathematics of Computation*, 48(117):329–339, January 1987.