Improved Stage 2 to $P \pm 1$ Factoring Algorithms

P. L. Montgomery and Alexander Kruppa

Microsoft Research, Redmond, USA and LORIA, Nancy, France

ANTS-VIII Banff, Canada, May 21st 2008

Contents

The P-1 factoring algorithm

- Introduced by Pollard (1974)
- Let N be odd, composite integer, prime $p \mid N$. Goal: find p
- Choose $b_0 \not\equiv \pm 1 \pmod{N}$, $\gcd(b_0, N) = 1$,
  and a highly composite integer e, e.g. $e =$ lcm$(1, 2, 3, \ldots, B_1)$
- Compute $b = b_0^e \bmod N$
- If $p - 1 \mid e$, then $b_0^e \equiv 1 \pmod{p}$ and $p \mid \gcd(b - 1, N)$
- Finds p quickly if $p - 1$ has only small prime factors (is “smooth”)

The P+1 factoring algorithm

- Introduced by Williams (1982)
- Works in $\mathbb{F}_{p^2}^*$, tries to construct element α of order $p+1$
- Computes $\alpha^e + \alpha^{-e}$ using Chebyshev polynomials $V_n(x + x^{-1}) = x^n + x^{-n}$
- Manipulating $\alpha^n + \alpha^{-n}$ allows arithmetic in base ring $\mathbb{Z}/N\mathbb{Z}$, we try to preserve this symmetry
- If $\alpha^e \equiv 1 \pmod{p}$, then $p \mid \gcd(\alpha^e + \alpha^{-e} - 2, N)$

The Stage 2

- What if stage 1 fails to find a factor?
- Could increase B_1, expensive
- Maybe $p - 1 = nq$ where $n \mid e$, q not too large (likely prime), then $b^q \equiv 1 \pmod{p}$. Try to find q up to B_2
- One stage 2 variant uses polynomial multipoint evaluation: degree d on n points in geometric progression with length $l = d + n$ convolution
- Number of q values tested: dn, so $B_2 \sim l^2$
- Reaches high B_2, needs much memory

The Stage 2 (cont.)

- Choose highly composite P, assuming $q \perp P$.
- Choose set S as full set of representatives of $(\mathbb{Z}/P\mathbb{Z})^*$
- Build $f(X) = \prod_{k \in S} \left(X - b^k\right) \bmod N$, degree $s = |S| = \varphi(P)$
- Evaluate all $f\left(b^{mP}\right) \bmod N$, $m_1 \leq m < m_2$
- If $q \perp P$, there is $k \in S$ so that $q = m'P - k$
- Hence $\left(b^{m'P} - b^k\right) \equiv b^k(b^q - 1) \equiv 0 \pmod{p}$ and we find $p \mid \gcd\left(f\left(b^{m'P}\right), N\right)$
- Includes q if $m_1 \leq m' < m_2$, effective $B_2 \approx m_2 P$

Previous work

Montgomery and Silverman (1990), “An FFT extension to the P-1 factoring algorithm”:

- Build $f(X)$ with product tree in $O(n \log(n)^2)$ multiplications
- Evaluate along geometric progression
- Mention in remarks that method extends to $P+1$, $f(X)$ can be built faster, reciprocal polynomials save space
- We implement these ideas

Previous work (cont.)

GMP-ECM, unified stage 2 for P–1, P+1, ECM:

- described in Zimmermann, Dodson (2006)
- mostly modeled after Montgomery's thesis (1992)
- product tree for $f(X)$
- general multipoint evaluation in $O(n \log(n)^2)$ time and $O(n \log(n))$ space

Our contribution

Our new stage 2

Factoring *S*

- For a given P, we want S a full set of representatives of $(\mathbb{Z}/P\mathbb{Z})^*$
- Set of sums $A + B = \{a + b, a \in A, b \in B\}$
- Factor S into sum of sets $T_1 + \ldots + T_n$ to speed up building $f(X)$
- Chinese Remainder Theorem: if $m \perp n$

$$(\mathbb{Z}/(mn)\mathbb{Z})^* = n(\mathbb{Z}/m\mathbb{Z})^* + m(\mathbb{Z}/n\mathbb{Z})^*$$

- Assume $4 \mid P$
- One set for each prime dividing P. Example: $P = 28$, $S = T_1 + T_2 = \{7, 21\} + \{4, 8, 12, 16, 20, 24\}$

Factoring S (cont.)

- More, smaller sets possible
- Let $R_n = \{2i - n - 1 : 1 \le i \le n\}$ be arithmetic progression of length n, common difference 2, symmetric around 0
- R_{p-1} is set of representatives of $(\mathbb{Z}/p\mathbb{Z})^*$, $p > 2$
- $R_{mn} = R_m + mR_n$, decompose into sets of prime cardinality
- One set for each prime in $\varphi(P)$. Example: $P = 28$, $S = T_1 + T_2 + T_3 = \{-7, 7\} + \{-16, 0, 16\} + \{-4, 4\}$
- All T_i symmetric around 0, so S symmetric around 0

Reciprocal Laurent Polynomials

- S symmetric around 0, $s = |S|$ even
- Make $f(X)$ reciprocal polynomial

$$f(X) = X^{-s/2} \prod_{k \in S, k>0} \left(X - b^k\right)\left(X - b^{-k}\right)$$
$$= \prod_{k \in S, k>0} \left(X - (b^k + b^{-k}) + X^{-1}\right)$$

- $f(X)$ monic reciprocal Laurent polynomial (RLP) of form $f(X) = f_0 + \sum_{i=1}^{s/2} f_i(X^i + X^{-i})$
- Only $s/2+1$ coefficients
- $P+1$: $\alpha^k + \alpha^{-k} = V_k(\alpha + \alpha^{-1})$, coefficients in base ring

Fast construction of *f(X)*

- Let $S = T_1 + T_2 + \ldots + T_n$, all T_i sets of prime cardinality, symmetric around 0. Assume $|T_n| = 2$
- We want

$$\begin{aligned} f(X) &= X^{-s/2} \prod_{k_1 \in S} (X - b^{k_1}) \\ &= X^{-s/2} \prod_{t_1 \in T_1} \prod_{t_2 \in T_2} \cdots \prod_{t_n \in T_n} (X - b^{t_1 + t_2 + \ldots + t_n}) \end{aligned}$$

- Compute right-to-left: start with $T_n = \{t_n, -t_n\}$,

$$f_n(X) = X^{-1}(X - b^{t_n})(X - b^{-t_n}) = X - (b^{t_n} + b^{-t_n}) + X^{-1}$$

- Expand for $i = n-1, \ldots, 1$:

$$f_i(X) = \prod_{t_i \in T_i} b^{t_i \deg(f_{i+1})} f_{i+1}(b^{-t_i} X)$$

Fast construction of *f(X)* (cont.)

- Then $f_1(X) = f(X)$
- If $|T_i| = 2$, $f_i(X)$ is product of polynomials of equal degree: efficient, do last. Sort T_i so that $|T_n| = 2$, $|T_i| \leq |T_i + 1|$ for $i = 1, \ldots, n-2$
- Complexity: $M(n)$ cost of polynomial multiplication. Many $|T_i| = 2$ so that cost only $\approx M(s/2) + M(s/4) + \ldots \leq M(s)$
- Scaled polynomial is not RLP, but $f_{i+1}(b^{t_i}X)f_{i+1}(b^{-t_i}X)$ is. Rewrite this product using Chebyshev polynomials to do all computations with RLPs over base ring

Multiplying RLPs

- Given reciprocal Laurent polynomials $Q(X)$, $R(X)$, we want $S(X) = Q(X)R(X) = s_0 + \sum_{i=1}^{d_s} s_i(X + X^{-1})$
- Monomial basis: $2d_s + 1$ terms, only $d_s + 1$ distinct coefficients, would like to use cyclic convolution of length $d_s + 1 \le l < 2d_s$.
- Computing $S(X)$ in monomial basis $\bmod X^l - 1$ does not work for $l < 2d_s$
- Example: $d_s = 3$, $l = 4$

x^{-3}	x^{-2}	x^{-1}	x^0	x^1	x^2	x^3	
s_3	s_2	s_1	s_0	s_1	s_2	s_3	
			s_0	$s_1 + s_3$	$s_2 + s_2$	$s_3 + s_1$	$\pmod{x^4 - 1}$

- Can't separate s_i, s_{l-i} for $i \ne 0, l/2$

Multiplying RLPs (cont.)

- Idea: use weighted convolution
- Multiply $\tilde{S}(wX) = Q(wX)R(wX) \bmod X^l - 1$:

x^{-3}	x^{-2}	x^{-1}	x^0	x^1	x^2	x^3
$w^{-3}s_3$	$w^{-2}s_2$	$w^{-1}s_1$	$w^0 s_0$	$w^1 s_1$	$w^2 s_2$	$w^3 s_3$
			$w^0 s_0$	$w^1 s_1 + w^{-3}s_3$	$w^2 s_2 + w^{-2}s_2$	$w^3 s_3 + w^{-1}s_1$

- After un-weighting $\tilde{S}(X)$:

x^0	x^1	x^2	x^3
s_0	$s_1 + w^{-4}s_3$	$s_2 + w^{-4}s_2$	$s_3 + w^{-4}s_1$

- If $w^l \neq 0, \pm 1$, we can separate s_i, s_{l-i}:

$$\left(w^l - w^{-l}\right) s_i = w^l \left(s_i + w^{-l}s_{l-i}\right) - \left(s_{l-i} + w^{-l}s_i\right)$$

Multipoint evaluation

- We want to evaluate RLP $f(X)$ at $X = b^{mP}$, $m_1 \leq m < m_2$
- Evaluating $f(X)$ at n points in geometric progression possible with convolution of length $l = \deg(f) + n$
- Most efficient if $\deg(f) \approx n$: choose $s = \varphi(P)$ close to $l/2$ for available transform lengths l
- Form $h(X) = f_0 + \sum_{j=1}^{s/2} f_j b^{-j^2 P}(X^j + X^{-j})$, an RLP
- $h(X)$ is reciprocal: $h(\omega^i) = h(\omega^{l-i})$ in length l DFT, ω an l-th root of unity: only $l/2 + 1$ distinct Fourier coefficients

Multipoint evaluation (cont.)

- Let $g(X) = \sum_{i=0}^{l-1} x_0^{M-i} b^{P(M-i)^2} X^i$, where $x_0 = b^{m_1 P}$, $M = l - 1 - s_1/2$
- Then coefficient of X^{M-i} in $g(X)h(X)$ is
$$x_0^m b^{Pm^2} \underline{f\left(b^{(m_1+i)P}\right)}$$
- For P+1: coefficients of $g(X)$, $h(X)$ in quadratic extension: need twice the memory, two convolutions

The algorithm: Summary

1. Choose P, S
2. Build $f(X)$ from factored S_1 (in $O(M(s))$)
3. Build $h(X)$ (in $O(l)$)
4. Build $g(X)$ (in $O(l)$)
5. Compute $g(X)h(X)$ (in $O(M(l))$)
6. Take gcd of coefficients and N (in $O(l)$). Print any factor

Our implementation

Our implementation: Timings and results

Our implementation

- Based on GMP-ECM, implementation of P-1, P+1, ECM
- Stage 1 unchanged from previous version
- Uses number-theoretic transform modulo small primes with CRT for convolutions, written by D. Newman, J. S. Papadopoulos
- On SMP: allows for parallelization, different cores process different primes
- Only power of 2 transform lengths so far

Our implementation: Timings

Time for stage 2 on 230 digit number with $B_2 = 1.2 \cdot 10^{15}$, $l = 2^{24}$, $s_1 = 7434240$, $s_2 = 3$ on 2.4GHz Opteron with 2 cores, 8GB:

	1 core	2 cores	
		cpu	elapsed
P–1	1738s	1753s	941s
P+1	3356s	3390s	2323s

For comparison, old P-1 stage 2: 34080s with 1 core (similar for P+1)

Time for stage 2 with $1.34 \cdot 10^{16}$, $l = 2^{26}$, $s_1 = 33177600$, $s_2 = 2$ on 2.6GHz Opteron with 32GB, 8 cores:

	8 cores	
	cpu	elapsed
P–1	5483s	922s
P+1	10089s	2192s

Results

Method	Number	factor size	q	size of q
P–1	$73^{109}-1$, c191	p50	462832247372839	15 digits

$p = 76227040047863715568322367158695720006439518152299$

Method	Number	factor size	q	size of q
P–1	$24^{142}+1$, c183	p53	12750725834505143	17 digits

$p = 20489047427450579051989683686453370154126820104624537$

Method	Number	factor size	q	size of q
P+1	$47^{146}+1$, c235	p52	843497917739	12 digits

$p = 7986478866035822988220162978874631335274957495008401$

Method	Number	factor size	q	size of q
P+1	L_{2366}, c290	p60	483576618980159	15 digits

$p = 725516237739635905037132916171116034279215026146021770250523$

60 digit factor set new record for P+1!

New GMP-ECM release

- New release of GMP-ECM version 6.2 implements new stage 2 for P–1, P+1
- Now available at

`http://gforge.inria.fr/projects/ecm/`

Brent-Suyama vs. large B_2

- Brent-Suyama extension: roots $b^{d(k_1)}$ and points of evaluation $b^{d(-k_2+mP)}$, e.g. $d(x) = x^{12}$, includes values $> B_2$

- New stage 2 allows larger B_2. Which is better?

- Example probabilities with $B_1 = 10^{11}$, typical parameters used by GMP-ECM:

	$B_2 = 10^{14}$		$B_2 = 2 \cdot 10^{14}$	$B_2 = 10^{15}$
		x^{120}		
$p \approx 10^{45}$	0.0179	0.0197	0.0196	0.0236
$p \approx 10^{50}$	0.0065	0.0071	0.0071	0.0087
$p \approx 10^{55}$	0.0022	0.0024	0.0024	0.0030

- Brent-Suyama extension improves probability of finding factors, but increasing B_2 by factor > 2 is better

Multi-point evaluation: Example

Example: $F(X) = f_0 + f_1 X + f_2 X^2$, evaluate at $X = c, cr, cr^2$

Let $h(X) = f_0 + f_1 cX/r + f_2 c^2 X^2/r^3$,
$g(X) = r^{10} + r^6 X + r^3 X^2 + rX^3 + X^4$

Then $g(x)h(x)$ is

$$
\begin{array}{c|lcccccc}
X^6 & & & & & f_2c^2/r^3 & & \\
X^5 & & & f_1c/r & + & f_2c^2/r^2 & & \\
X^4 & f_0 & + & f_1c & + & f_2c^2 & = & F(c) \\
X^3 & f_0r & + & f_1cr^2 & + & f_2c^2r^3 & = & rF(cr) \\
X^2 & f_0r^3 & + & f_1cr^5 & + & f_2c^2r^7 & = & r^3F(cr^2) \\
X^1 & f_0r^6 & + & f_1cr^9 & & & & \\
X^0 & f_0r^{10} & & & & & &
\end{array}
$$