MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

BETTER POLYNOMIALS FOR GNFS

SHI BAI, CYRIL BOUVIER, ALEXANDER KRUPPA, AND PAUL ZIMMERMANN

Abstract. The general number field sieve (GNFS) is the most efficient algorithm known for factoring large integers. It consists of several stages, the first one being polynomial selection. The quality of the selected polynomials can be modelled in terms of size and root properties. We propose a new kind of polynomials for GNFS: with a new degree of freedom, we further improve the size property. We demonstrate the efficiency of our algorithm by exhibiting a better polynomial than the one used for the factorization of RSA-768, and a polynomial for RSA-1024 that outperforms the best published one.

1. Introduction to GNFS

The general number field sieve [11] is the most efficient algorithm known for factoring large integers. It has been used in many record factorizations such as RSA-704 [3] and RSA-768 [18]. GNFS consists of several stages including polynomial selection, sieving, filtering, linear algebra and finding square roots.

Let n be the integer to be factored. In polynomial selection, we want to choose two irreducible and coprime polynomials $f(x)$ and $g(x)$ over $\mathbb{Z}$ that share a common root m modulo n. In practice, the homogenized polynomials $F(x, y)$ and $G(x, y)$ are often used. We want to find many coprime pairs $(a, b) \in \mathbb{Z}^2$ such that the polynomials values $F(a, b)$ and $G(a, b)$ are simultaneously smooth with respect to some bounds B_1 and B_2. An integer is smooth with respect to bound B (or B-smooth) if none of its prime factors are larger than B. The line sieving and lattice sieving [17] are commonly used to identify such pairs (a, b).

The running-time of sieving depends on the quality of the chosen polynomials in polynomial selection, hence many polynomial pairs will be generated and optimized in order to produce a good one.

This paper proposes a new kind of polynomials for the number field sieve, together with a corresponding algorithm to generate such polynomials, and exhibits better polynomials found by this algorithm for RSA challenge numbers.

2. Polynomial selection

For large integers, most methods for polynomial selection [6, 9, 10, 12, 13] in GNFS use a linear polynomial for $g(x)$ and a nonlinear one for $f(x)$ (degree 6 in latest records). The standard method to generate such polynomial pairs is to expand n in base-(m_1, m_2) so $n = \sum_{i=0}^{d} c_i m_1^i m_2^{d-i}$. The polynomial pair is given by $f(x) = \sum_{i=0}^{d} c_i x^i$ and $g(x) = m_2 x - m_1$.

2010 *Mathematics Subject Classification.* Primary 11Y05, 11Y16.

The first author was supported in part by the ERC Starting Grant ERC-2013-StG-335086-LATTAC.

The running-time of sieving depends on the smoothness of the polynomial values $|F(a,b)|$ and $|G(a,b)|$. Let $\Psi(z, z^{1/u})$ be the number of $z^{1/u}$-smooth integers below z for some $u > 0$. The Dickman-de Bruijn function $\rho(u)$ [7] is often used to estimate the density of smooth numbers $\Psi(z, z^{1/u})$. It can be shown that

$$\lim_{z\to\infty} \frac{\Psi(z, z^{1/u})}{z} = \rho(u).$$

The Dickman-de Bruijn function satisfies the delay-differential equation

$$u\rho'(u) + \rho(u-1) = 0, \quad \rho(u) = 1 \text{ for } 0 \le u \le 1.$$

It may be shown that ρ satisfies the asymptotic estimate

$$\log(\rho(u)) = -(1+o(1))u\log u \text{ as } u \to \infty.$$

For practical purposes, the frequency of smooth numbers can be approximated by the Canfield-Erdős-Pomerance theorem, which can be stated as follows (Corollary 1.3 from [8]):

Theorem 2.1. *For any fixed $\epsilon > 0$, we have*

$$\Psi(z, z^{1/u}) = zu^{-u(1+o(1))}$$

as $z^{1/u}$ and u tend to infinity, uniformly in the region $z \ge u^{u/(1-\epsilon)}$.

We want to choose polynomials that produce many smooth polynomial values across the sieve region. This heuristically requires that the size of polynomial values is small in general. In addition, one can choose an algebraic polynomial $f(x)$ which has many roots modulo small prime powers. Then the polynomial values are likely to be divisible by small prime powers. This may increase the smoothness probability of polynomial values. In the rest of this section, we recall some methods [9, 13] to estimate and compare the quality of polynomials.

2.1. **Quality of polynomials.** The quality of the chosen polynomials in polynomial selection can be modelled in terms of size and root properties [13].

2.1.1. *Size property.* Let (a,b) be pairs of relatively prime integers in the sieving region Ω. Since G is a linear polynomial, we may assume that $\log(|G(a,b)|)$ does not vary much across the sieving region. We thus only consider the size of the nonlinear polynomial F, which is modelled by the circular logarithmic L^2-norm (the smaller the better):

$$\frac{1}{2}\log\left(s^{-d}\int_0^{2\pi}\int_0^1 F^2(s\cos\theta, \sin\theta)\, r^{2d+1}\,\mathrm{d}r\,\mathrm{d}\theta\right). \tag{2.1}$$

For the norm defined in Equation (2.1), one should not only be able to estimate accurately that norm for a given skewness s, but find the optimal skewness that gives the minimal norm.

For a sextic polynomial, the logarithmic L^2-norm in Equation (2.1) can be expressed as follows, where $\tilde{c_i} = c_i s^{i-d/2}$:

$$\begin{aligned}\frac{1}{2}\log\Big(\frac{\pi}{7168}\big(&231\,\tilde{c}_0^2 + 42\,\tilde{c}_0\tilde{c}_2 + 14\,\tilde{c}_0\tilde{c}_4 + 10\,\tilde{c}_0\tilde{c}_6 + 21\,\tilde{c}_1^2 + 14\,\tilde{c}_1\tilde{c}_3\\ &+ 10\,\tilde{c}_1\tilde{c}_5 + 7\,\tilde{c}_2^2 + 10\,\tilde{c}_2\tilde{c}_4 + 14\,\tilde{c}_2\tilde{c}_6 + 5\,\tilde{c}_3^2 + 14\,\tilde{c}_3\tilde{c}_5\\ &+ 7\,\tilde{c}_4^2 + 42\,\tilde{c}_4\tilde{c}_6 + 21\,\tilde{c}_5^2 + 231\,\tilde{c}_6^2\big)\Big).\end{aligned} \tag{2.2}$$

For a degree-d polynomial $f(x)$, locating the extrema with respect to s of the term inside the logarithm simply reduces to finding the roots of a degree-d polynomial, and keeping the root giving the smallest norm. We often say "the skewness of f" for this optimal skewness s.

2.1.2. *Root property.* If a polynomial $f(x)$ has many roots modulo small prime powers, the polynomial values may behave more smooth than random integers of about the same size. Boender, Brent, Montgomery and Murphy [5, 12, 13, 14] described some quantitative measures of this effect (root property). Let n_{p^k} be the number of roots of $f \pmod{p^k}$ for $k \geq 1$. The expected p-valuation $\nu_p(f)$ is defined to be $\nu_p(f) = \sum_{k=1}^{\infty} n_{p^k}/(p^{k-1}(p+1))$. The root property can be quantified by

$$\alpha(F) = \sum_{\substack{p \leq B \\ p \text{ prime}}} \left(\frac{1}{p-1} - \nu_p(F) \right) \log p,$$

which compares the cumulative expected p-valuation of polynomial values to random integers of similar size. We refer to [2, 4, 13] for more details.

2.1.3. *Murphy's* E *score.* Murphy's E score [13] is a (relatively) reliable ranking function to identify the best polynomials without test sieving. Taking the root property into account, the number of sieving reports (coprime pairs that lead to smooth polynomial values) can be approximated by

$$\frac{6}{\pi^2} \int_{\Omega} \rho\left(\frac{\log|F(x,y)| + \alpha(F)}{\log B_1} \right) \rho\left(\frac{\log|G(x,y)| + \alpha(G)}{\log B_2} \right) \mathrm{d}x\, \mathrm{d}y.$$

For comparison, one can ignore the constant multiplier $6/\pi^2$. To approximate the integral, Murphy used a summation over a set of K sample points (x_i, y_i):

$$\mathrm{E}(F,G) = \sum_{i=1}^{K} \rho\left(\frac{\log|F(x_i,y_i)| + \alpha(F)}{\log B_1} \right) \rho\left(\frac{\log|G(x_i,y_i)| + \alpha(G)}{\log B_2} \right).$$

Over a circular region of radius r, we can sample $x_i = r\cos\theta_i$ and $y_i = r\cos\theta_i$. The angles θ_i sample the points on (the boundary of) the circular region. The Dickman-de Bruijn function $\rho(x)$ does not admit a closed form solution. An asymptotic expansion can be used to approximate its values. Murphy's E score is a better ranking function and we use it in later sections to compare polynomials.

2.2. **Optimizing the quality of polynomials.** Polynomial selection can be divided into three steps: polynomial generation, size optimization and root optimization.

In polynomial generation, we generate many raw polynomials whose size is admissible. We further reduce the size of the raw polynomials in size optimization. Many polynomials can have comparable size after size optimization. We produce and choose the best polynomials in terms of Murphy's E score in root optimization.

Translation and rotation are useful to optimize the size and root properties. Let $f(x) = \sum_{i=0}^{d} c_i x^i$ and $g(x) = m_2 x - m_1$ where $m_1/m_2 \pmod n$ is the common root.

Translation of $f(x)$ by k gives a new polynomial $f(x+k)$; the linear polynomial becomes $g(x) + km_2$, and the common root is changed to $m_1/m_2 - k \pmod n$. Translation does not alter the root properties.

Rotation by a polynomial $\lambda(x)$ gives a new polynomial $f(x) + \lambda(x)\,g(x)$; the linear polynomial and the common root are unchanged. $\lambda(x)$ is often a linear or quadratic polynomial, depending on n and on the skewness of $f(x)$. Rotation can affect both size and root properties.

The contributions of this paper are threefold: (i) we introduce a new kind of polynomials for GNFS; (ii) we propose an algorithm which generates such polynomials, assuming a good translation k is known; and (iii) we propose an algorithm to find such a good translation.

3. Size optimization

Polynomial generation (e.g., using Kleinjung's methods [9, 10]) gives many raw polynomials with small leading coefficients. The raw polynomials have very small $|c_d|, |c_{d-1}|$ and small $|c_{d-2}|$. The coefficients $|c_{d-3}|, \cdots, |c_0|$ are comparable to $m_1 \approx (n/c_d)^{1/d}$. In size optimization, we want to produce polynomials with smaller logarithmic L^2-norm (e.g., Equation (2.1)) by changing the skewness, translating and rotating.

Assuming no cancellation occurs, we can approximate $|F(a,b)| \approx \sum_{i=0}^{d} |c_i a^i b^{d-i}|$; this approximation is maximal at the corner of the sieve region $a = U\sqrt{s}$, $b = U/\sqrt{s}$, where $|F(a,b)| \approx U^d \sum_{i=0}^{d} |c_i s^{i-d/2}|$. For quintic polynomials, $|c_5|, |c_4|$ and $|c_3|$ are small when using Kleinjung's algorithm. The next non-controlled coefficient is c_2. As $s \geq 1$, the dominant term is $|c_2| s^{-1/2} U^5$, so the contribution of c_2 on the polynomial value is already reduced by a factor of $s^{-1/2}$.

For sextic polynomials, the approximate polynomial values are dominated by the term $|c_3|U^6$ in the regions where no cancellation occurs. Here, c_3 is not controlled in the polynomial generation step, and we do not get a reduction in size like the $s^{-1/2}$ factor for quintic polynomials. Therefore, it is important to size-optimize sextic polynomials before trying to optimize the root properties.

Sextic polynomials are of main interest since they have been used in record factorizations such as RSA-768 [18] and should be used for future records. Murphy [13] shows that the running time of the number field sieve, to factor an integer n with a degree-d polynomial, is about

$$\exp\left((1+o(1))\left(d\log d + \sqrt{(d\log d)^2 + 4\log(n^{1/(d+1)})\log\log(n^{1/(d+1)})}\right)\right).$$

With numerical calculations for various n and d, this would indicate that sextic polynomials are preferable for numbers between 220 and 360 decimal digits. The two challenge numbers RSA-896 (270 digits) and RSA-1024 (309 digits) are thus suitable for using sextic polynomials.

Let $f(x)$ be a sextic polynomial. We can use quadratic rotations since $c_3, \ldots, c_0$ have order $(n/c_6)^{1/6}$. A quadratic rotation is defined for integers u, v, w by:

$$f_{u,v,w}(x) = f(x) + (ux^2 + vx + w)\,g(x). \tag{3.1}$$

3.1. Some "classical" methods.

We describe here some of the state-of-the art size-optimization ideas, some of which are not due to the authors of this article, but since they are not published elsewhere, we mention them for completeness.

We want to produce a polynomial with small L^2-norm by translation and rotation. We focus on degree 6 here for sake of clarity, it is straightforward to generalize to other degrees.

In the raw polynomial, c_0, c_1, c_2, c_3 have similar size and are much larger than c_4, c_5, c_6. In Equation (2.2), $\tilde{c}_0, \tilde{c}_1, \tilde{c}_2$ are bounded by $\tilde{c}_3$. Therefore, the L^2-norm can be controlled by terms involving $\tilde{c}_3, \tilde{c}_4, \tilde{c}_6$ (since $|c_6| \approx |c_5| \ll |c_4|$). Assuming no cancellation occurs in Eq. (2.2), a lower bound, not depending on skewness, is the term $\tilde{c}_3^2 = c_3^2$. Hence, a small c_3 is a necessary condition for a small L^2-norm. The idea is to minimize c_3 by translation. Translation by k gives a polynomial in x whose coefficients are functions of k:

$$f(x+k) = c_6x^6 + (6c_6k + c_5)x^5 + (15c_6k^2 + 5c_5k + c_4)x^4 + (20c_6k^3 + 10c_5k^2 + 4c_4k + c_3)x^3 + \cdots.$$

Let $c_i(k)$ be the coefficients of the i-th term in the translated polynomial. $c_3(k)$ of $f(x+k)$ is a cubic polynomial in k. The coefficients $c_0(k), c_1(k), c_2(k)$ will increase due to translation. We can use rotation to reduce them, if needed.

3.1.1. *Minimizing $c_3(k)$.* The cubic polynomial $c_3(k)$ has either one or three real roots (this is not particular to degree 6: the coefficient $c_{d-3}(k)$ is always cubic in k). For each real root r, we choose k_0 to be either $\lceil r \rceil$ or $\lfloor r \rfloor$, whichever minimizes $|c_3(k)|$. We translate $f(x)$ by k_0. Then we can further optimize the polynomial by a local descent method.

We give an example with the raw polynomial of A_{768} from §4.1. The coefficient of x^3 in $f(x+k)$ is

$$1810534320k^3 + 2410120740k^2 - 1189404920661858930542720k + 729479045157502847705046405886586876 4.$$

This cubic polynomial has a real root near $k_0 = -1591376391$. We translate $f(x)$ by k_0, which reduces the coefficient c_3 from $7.3 \cdot 10^{36}$ to $-1.4 \cdot 10^{27}$. After reducing the coefficients of degree $2, 1, 0$ of $f(x + k_0)$ by rotation, we obtain a logarithmic L^2-norm of 77.36, compared to 81.82 for $f(x)$. We then apply a local optimization method.

This method works better on average than a local optimization used alone. Table 1 (see also Figure 1) shows that on our RSA-768 data set of 10^5 polynomials, it reduces the average logarithmic L^2-norm to 69.84 (with standard deviation 0.56).

3.2. **A new class of GNFS polynomials.** Previous polynomial selection algorithms that generate a nonlinear polynomial $f(x)$ and a linear one $g(x)$ all produced polynomials such that $|\mathrm{Res}(f, g)| = n$ where n is the number to factor.

We propose a more general class, such that $|\mathrm{Res}(f, g)| = \ell n$, where ℓ is a small integer, which we call the "multiplier", as in the MPQS polynomial selection [19]. We describe a lattice-based algorithm to find such polynomials.

3.3. **An algorithm to find such polynomials.** For the sake of simplicity we describe the algorithm for a degree-6 polynomial, but it applies to any degree d-polynomial, as long as $d \geq 3$.

Given a raw degree-6 polynomial $f(x) = c_6x^6 + \cdots + c_1x + c_0$, and a linear polynomial $g(x) = m_2x - m_1$ (for example found by Kleinjung's algorithms [9, 10]), we want to find a polynomial of small L^2-norm, as defined in Eq. (2.1), by using translation and rotation. Choose a positive integer s (corresponding to the skewness of f) and form the following lattice from the coefficients of f (each column

representing a vector):

$$L = \begin{pmatrix} s^6c_6 & 0 & 0 & 0 & 0 \\ s^5c_5 & 0 & 0 & 0 & 0 \\ s^4c_4 & s^4m_2 & 0 & 0 & 0 \\ s^3c_3 & -s^3m_1 & s^3m_2 & 0 & 0 \\ s^2c_2 & 0 & -s^2m_1 & s^2m_2 & 0 \\ sc_1 & 0 & 0 & -sm_1 & sm_2 \\ c_0 & 0 & 0 & 0 & -m_1 \end{pmatrix}.$$

Using the Lenstra-Lenstra-Lovász algorithm (LLL), one obtains short vectors of the form:

$$\begin{pmatrix} s^6(\ell c_6) \\ s^5(\ell c_5) \\ s^4(\ell c_4 + tm_2) \\ s^3(\ell c_3 - tm_1 + um_2) \\ s^2(\ell c_2 - um_1 + vm_2) \\ s(\ell c_1 - vm_1 + wm_2) \\ \ell c_0 - wm_1 \end{pmatrix}$$

which corresponds to the polynomial $\ell f(x)+(tx^3+ux^2+vx+w)g(x)$, after dividing (exactly) the coefficients by powers of s. Then one applies some local optimization procedure. If one wants the first vector of L to be not much larger than the last one in norm, one will take $s \leq (m_1/c_6)^{1/6}$.

In practice the polynomial $\ell f(x) + (tx^3 + ux^2 + vx + w)g(x)$ corresponding to the shortest vector is the same for a large range of skewness s; it thus suffices to consider a few values of s (for example $s = 10^3, 10^4, 10^5, 10^6$).

This LLL-based algorithm finds a good rotation $\ell f(x)+(\cdots+ux^2+vx+w)g(x)$. To find a good translation we apply this algorithm on $f(x + k)$ for several values of k, and keep the overall best polynomials (in §3.4, we describe how to find good values of the translation k).

Previous work has focused on $\ell = 1$ only; here we allow the *multiplier* $\ell > 1$. If $|\mathrm{Res}(f, g)| = n$ where n is the number to factor, then with a multiplier we get $|\mathrm{Res}(\ell f+\cdots, g)| = \ell n$. Thus the resultant is increased by a factor ℓ, but nonetheless we might find better polynomials (as demonstrated in §3.5 and §4).

It should be noted that one does not try to reduce the coefficient c_5, since with Kleinjung's algorithm, it is of the same order of magnitude as c_6.

This method works better on average than the previous methods. Table 1 shows that it reduces the average logarithmic L^2-norm to 68.42 (with standard deviation 0.72) on our RSA-768 data set of 10^5 polynomials.

3.4. **Finding a good translation.** In this section we describe two methods to generate translations that can be used as input for the algorithm described in §3.3.

3.4.1. *Minimizing $c_{d-2}(k)$ and $c_{d-3}(k)$ simultaneously.* We describe an improvement over the method in §3.1.1.

After translation and the algorithm described in §3.3, if only the rotation of highest degree is considered, the new polynomial is in form of $\ell f(x+k)+tx^{d-3}g(x+k)$, where ℓ and t are integers. Let $q = t/\ell$, the problem reduces to finding rational values of q for which there exists values of k that make $\hat{c}_{d-2}(k)$ and $\hat{c}_{d-3}(k)$

small in $\hat{f}(x) = f(x+k) + qx^{d-3}g(x+k) =: \sum_{i=0}^{d} \hat{c}_i(k)x^i$. As we want to minimize $\hat{c}_{d-2}(k)$ and $\hat{c}_{d-3}(k)$ simultaneously, we try to find values of q for which $\mathrm{Res}_k(\hat{c}_{d-3}(k), \hat{c}_{d-2}(k))$ has a root or is small (in absolute value).

For degree-6 polynomials, $\hat{c}_{d-2}(k)$ and $\hat{c}_{d-3}(k)$ have the following form, with $g = m_2x - m_1$:

$$\hat{c}_4(k) = 15c_6k^2 + 5c_5k + c_4 + qm_2;$$
$$\hat{c}_3(k) = 20c_6k^3 + 10c_5k^2 + 4c_4k + qm_2k + c_3 - qm_1.$$

Then $\mathrm{Res}(\hat{c}_3(k), \hat{c}_4(k))$ is a polynomial of degree 3 in q, with integer coefficients. So it has either 1 or 3 real roots.

In the case where the resultant has 3 real roots, each root can be approximated by a rational number, using continued fractions, Farey approximations or trying all rational fractions with bounded denominator. For each of these rational approximations, roots of $\hat{c}_4(k)$ and $\hat{c}_3(k)$ can be computed (they are close, by choice of q), and can be rounded to obtain values for the translation k.

In the case where the resultant has 1 real root, experiments show that it is necessary to consider also extrema of the resultant and not only the root. Then, as in the other case, rational approximations of these values of q are used to find values of k.

In both cases, the values of k found provide pairs of translated polynomials $(f(x+k), g(x+k))$ that can be used as input for the algorithm described in §3.3.

3.4.2. *Translations of the form $i \times 10^j$.* In addition to the above direct method, one can try values of k of the form $i \cdot 10^j$ for small i, j. In practice, this helps to locate a few good values of k which are missed by the above algorithm.

3.5. **Experiments.** We examine a data set consisting of 10^5 raw sextic polynomials for RSA-768. Those polynomials were generated by CADO-NFS [1] and Msieve [16, 15] using Kleinjung's algorithm [10]. Table 1 (left) shows the average logarithmic L^2-norm for the raw and optimized polynomials. Figure 1 shows the normalized discrete density distribution of logarithmic L^2-norm for the raw and optimized polynomials.

TABLE 1. Comparison of size optimization methods on 10^5 raw polynomials for RSA-768 (left) and 5795 raw polynomials for RSA-512 (right). Columns $\log(L^2)$ and std. $\log(L^2)$ record the average logarithmic L^2-norm of polynomials and its standard deviation.

RSA-768	$\log(L^2)$	std. $\log(L^2)$
Raw polynomials	80.75	1.00
Method of §3.1.1	69.84	0.56
Method of §3.3	68.42	0.72

RSA-512	$\log(L^2)$	std. $\log(L^2)$
Raw polynomials	53.54	1.96
Method of §3.1.1	50.43	1.49
Method of §3.3	49.36	0.62

We performed a similar experiment with degree-5 raw polynomials produced by CADO-NFS for RSA-512 (see Table 1, right). Although the gain on the average logarithmic L^2-norm is smaller than for degree 6, it demonstrates the new algorithm is still valuable for degree 5.

In Table 2, we further consider some experiments to compare the final polynomials (after size and root optimization) obtained by the new method (c.f. §3.3) and the method in §3.1.1. For the RSA-768 data, we optimize the size of the raw polynomials in both ways and then run the root optimization on the size-optimized

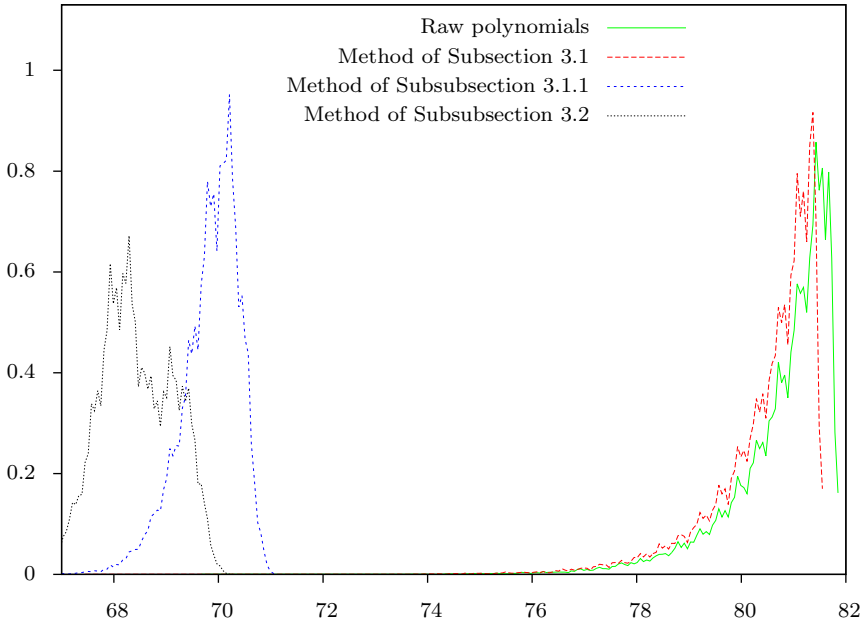

FIGURE 1. Distribution of logarithmic L^2-norm after size optimization.

polynomials. Assuming that Murphy's E score provides an accurate estimation of the yield rate, we can see that the new method produces polynomials with much higher yield rates on average.

TABLE 2. Comparison of two size optimization methods on 10^5 raw polynomials for RSA-768. Columns $\log(L^2)$ and $\alpha(F)$ record the average logarithmic L^2-norm and α-score of polynomials after size and root optimization; Column E is the average Murphy's E score (c.f. §2.1.3) for polynomials after size and root optimization; Column Top E is the average Murphy's E score for the top 100 polynomials. Here we use $B_1 = 1.1 \times 10^9$, $B_2 = 2.0 \times 10^8$ and the area 2.362×10^{18} for the domain Ω in the computation of Murphy's E score.

	$\log(L^2)$	$\alpha(F)$	E	Top E
Method of §3.1.1	71.86	-7.019	8.60e-14	2.14e-13
Method of §3.3	69.90	-6.812	1.10e-13	2.53e-13

4. Polynomials for RSA challenge numbers

In this section, we give polynomials for some RSA challenge numbers found by our new method, some of which have better Murphy's E score than the previously reported or published polynomials.

4.1. **RSA-768.** Using our algorithm on a data-set kindly provided by Jason Papadopoulos, we found several polynomials for RSA-768 that are better than the one used for its factorization, see Table 3.

TABLE 3. Better polynomials for RSA-768.

	Polynomial A_{768}: $\log(L^2) = 65.40$, E = 4.42e-13.
m_2	3653258925429788683931
m_1	154477669109765136712756724037850686 26
c_6	3258961776
c_5	288664131841057800
c_4	1103050623773707 4466307
c_3	-893188977600857037294644587163
c_2	943910582396304678841433664831 4151
c_1	377093715995883343269663077625960978403307
c_0	-904516168995007172662900583473883296530585 4530
	Raw polynomial of A_{768}: $\log(L^2) = 81.82$.
m_2	3653258925429788683931
m_1	154477669649080444719058876631998545 37
c_6	90526716
c_5	241012074
c_4	-297351230165464732635680
c_3	729479045157502847705046405886586 8764
c_2	283452995840471562081987321376267 5365
c_1	288524965019008859802888800564521 1453
c_0	651895590880755556906420587188754 8883
	Polynomial B_{768}: $\log(L^2) = 64.39$, E = 4.52e-13.
m_2	5924452599136152496277
m_1	305711325779271236010487446723983 40686
c_6	22604400
c_5	33946122310442580
c_4	74850428174211171973801
c_3	-25318719430823718613440606 4742
c_2	-81310927091457333751052543066797504
c_1	28772541579434794385398996592471406 4839458
c_0	-81187709244683044191049418357655772035310 11465
	Raw polynomial of B_{768}: $\log(L^2) = 82.44$.
m_2	5924452599136152496277
m_1	305711340607666944191907383959784361 48
c_6	1506960
c_5	-2418388
c_4	-1408315672283641690514244
c_3	-1426558909376529949901656712434177 2705
c_2	-97814534121904016489335854969382238 91
c_1	68698141669167832949898124129634274 66
c_0	116568343615941504924209811923882453 65

The polynomials A_{768} and B_{768} correspond to a multiplier $\ell = 36$ and 15 from their respective raw polynomials, thus could not have been found with previously known methods. The original polynomial used for the RSA-768 factorization has logarithmic-L^2 norm 64.08 and Murphy's E score 4.28e-13. Both polynomials A_{768} and B_{768} have better Murphy's E score. We use $B_1 = 1.1 \times 10^9$, $B_2 = 2.0 \times 10^8$ and the area 2.362×10^{18} in the computation of Murphy's E score.

A sieving test in the special-q range of length 10^5 starting at $3,400,000,000$ shows that the polynomial A_{768} produces 7% more relations per special-q than the original RSA-768 polynomial, and 5% more relations per second. For this test, we use the same binary than the one used for the factorization of RSA-768, with same

parameters. A similar sieving test shows that B_{768} produces 5% more relations per special-q than the original polynomial, and 3% more relations per second.

4.2. **RSA-896.** We consider a set of ten raw polynomials for RSA-896, which are deduced from c_6, m_2, m_1 using the algorithm in [9, Lemma 2.1]:

#	c_6	m_2	m_1
1	120	305986799480737271146945 67	38840974036761151681900363255247341949495 9325
2	120	39584252255977153653238969	38840974036091471118393830167343768411290 3927
3	180	24122614381393211892378929	36302920888345037533594763993085265139635 4800
4	240	127202957198327749843027	34603373980760708268993210773289788832928 6672
5	300	205701125136203480631 4333	33340090751470079438193623545241468481416 3915
6	480	54034135845123718658507 51	30828101522793436115065585124124690098215 8569
7	540	20178849932469701432255 89	30228831523556724404048084094917851630760 8795
8	600	55808326686191457067	29702644113166639147887049434519093950215 2703
9	600	52318705091858802318954527	29702644113374832823658042126568041701934 5427
10	600	10352691606130810454808797 3	29702644113522571092179926003616226729306 8705

We give for each one the logarithmic L^2-norm of the raw polynomial, the best size-optimized polynomial found with CADO-NFS 2.1 [1], and with our new algorithm:

#	1	2	3	4	5	6	7	8	9	10
Raw polynomial	98.28	98.11	96.89	98.00	97.84	98.53	97.18	98.37	96.97	96.63
CADO-NFS 2.1	82.88	82.74	82.30	82.03	82.37	83.33	82.12	79.36	83.79	82.45
New algorithm	80.53	80.16	79.33	79.75	79.78	79.83	80.04	80.72	79.92	79.38

The new algorithm yields a logarithmic L^2-norm which is smaller by 2.40 on average (79.94 against 82.34), and always smaller, with the exception of #8.

4.3. **RSA-1024.** Let us consider the degree-6 polynomial for RSA-1024 from [20, Appendix A]. This polynomial has logarithmic L^2-norm 100.02 according to Eq. (2.1). We ran our algorithm to re-optimize this polynomial, and obtained the polynomial A_{1024} of logarithmic L^2-norm 94.91: Therefore we can expect polynomial values $F(a, b)$ to be smaller by a factor about $\exp(100.02 - 94.91) \approx 166$.

Moreover, using our implementation in CADO-NFS, we found the polynomial B_{1024} which corresponds to a multiplier $\ell = 5$ from its raw polynomial. According to its Murphy's E score of 8.86e-12, against 9.75e-13 for the polynomial from [20], we can expect a relation yield about 9.1 times larger. With same parameters ($B_1 = B_2 = 10^{11}$, area $= 10^{18}$), we get a Murphy E score of 3.56e-09 for the polynomial used to factor RSA-768, which would give a sieving time for RSA-1024 about 402 times larger than for RSA-768 (instead of 1000 times as claimed in [18]). Note this polynomial was found in a thousand cpu hours, we expect a much better polynomial with a real search of a few thousands cpu years. This polynomial is also better than the one from [21], which has Murphy's E score 6.79e-12.

5. Conclusion

We introduced a new class of polynomials for GNFS, proposed new algorithms that produce such polynomials of small size, and demonstrated the efficiency of those algorithms on RSA challenge numbers. Those algorithms are implemented in CADO-NFS [1], an open-source implementation of the number field sieve.

TABLE 4. Two polynomials for RSA-1024.

	Polynomial A_{1024}: $\log(L^2) = 94.91$.
m_2	1
m_1	629042860635589902725572332002739172216328869941 3
c_6	1173597989242921482240
c_5	-436081570202935700372728737578556 16
c_4	691958140341173987625035104743657545537
c_3	450511202161208734370957748132330118597 3519
c_2	-173044525194396434037555851105075127649352575 00
c_1	-283131007738513042381019627129255517197411658676 33
c_0	249963295649448077896029171367943737823083959799485325
	Polynomial B_{1024}: $\log(L^2) = 91.90$, E = 8.86e-12.
m_2	2387707688882042760409842 1
m_1	333256330075525330759650655917856625450820494973 8
c_6	492999999999872400
c_5	19986130996295579328005858 00
c_4	14776348389733418096949161617663667
c_3	-173695632967027892479424675727980154323516
c_2	-582451394818326241473231984414006567833487818962
c_1	296096357723016232482734280196889286209855216805082715 6
c_0	-203645588998685384208162058984744030746414525938936824515 4065

ACKNOWLEDGEMENTS

The authors are grateful to Richard Brent, Nicholas Coxon, Steven Galbraith, Jean-Charles Gilbert, Thorsten Kleinjung, Jason Papadopoulos and Emmanuel Thomé for helpful comments and discussions. The RSA-768 data set provided by Jason Papadopoulos was obtained with the help of several contributors, among which Greg Childers, Lionel Debroux, Bruce Dodson, Jeff Gilchrist, Ed Hall, Jayson King, Alexander Kruppa, Paul Leyland, Luigi Morelli and a few other anonymous contributors.

The first author would like to thank the Australian National University, the University of Auckland and NeSI (New Zealand eScience Infrastructure) for providing funding and computing facilities.

Finally the authors are very grateful to the anonymous referee who asked for a major revision of the paper, in the process of which the new algorithm in §3.3 was discovered.

REFERENCES

1. S. Bai, C. Bouvier, A. Filbois, P. Gaudry, L. Imbert, A. Kruppa, F. Morain, E. Thomé, P. Zimmermann. CADO-NFS, an implementation of the number field sieve. Release 2.1, available from `http://cado-nfs.gforge.inria.fr`, 2014.
2. S. Bai, R. P. Brent, E. Thomé. Root optimization of polynomials in the number field sieve. Mathematics of Computation (to appear), 2014.
3. S. Bai, E. Thomé, P. Zimmermann. Factorisation of RSA-704 with CADO-NFS. Report, 2012. `http://eprint.iacr.org/2012/369.pdf`.
4. R. Barbulescu and A. Lachand. Some mathematical remarks on the polynomial selection in NFS. Report, 2014. `http://arxiv.org/abs/1403.0184`.
5. H. Boender. *Factoring large integers with the quadratic sieve*. PhD thesis, Leiden University, 1997.
6. J. Buhler, H. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. In Lenstra and Lenstra [11], pages 50–94.

7. A. Granville. Smooth numbers: computational number theory and beyond. In *Proc. MSRI Conf. Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography*. MSRI Publications, Volume 44, 2008.
8. A. Hildebrand and G. Tenenbaum. Integers without large prime factors. *Journal de Théorie des Nombres de Bordeaux*, 5(2):411–484, 1993.
9. T. Kleinjung. On polynomial selection for the general number field sieve. *Mathematics of Computation*, 75(256):2037–2047, 2006.
10. T. Kleinjung. Polynomial selection. In *CADO workshop on integer factorization*, INRIA Nancy, 2008. `http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf`.
11. A. K. Lenstra and H. W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, 1993.
12. B. A. Murphy. Modelling the Yield of Number Field Sieve Polynomials. In *Algorithmic Number Theory - ANTS III, LNCS 1443*, pages 137–147, 1998.
13. B. A. Murphy. *Polynomial selection for the number field sieve integer factorisation algorithm*. PhD thesis, The Australian National University, 1999.
14. B. A. Murphy and R. P. Brent. On quadratic polynomials for the number field sieve. In *Proceedings of the CATS '98*, volume 20 of *Australian Computer Science Communications*, pages 199–213. Springer, 1998.
15. J. Papadopoulos. Call for volunteers: RSA768 polynomial selection, 2011. `http://www.mersenneforum.org/showthread.php?t=15540`.
16. J. Papadopoulos. Msieve v1.48, 2011. `http://sourceforge.net/projects/msieve`.
17. J. M. Pollard. The lattice sieve. In Lenstra and Lenstra [11], pages 43–49.
18. T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. J. J. te Riele, A. Timofeev, and P. Zimmermann. Factorization of a 768-bit RSA modulus. In *Proceedings of CRYPTO '10*, volume 6223 of *Lecture Notes in Computer Science*, pages 333–350. Springer, 2010.
19. R. D. Silverman. The Multiple Polynomial Quadratic Sieve. *Mathematics of Computation*, 48(177):329–339, 1987.
20. A. Lenstra, E. Tromer, A. Shamir, W. Kortsmit, B. Dodson, J. Hughes, and P. Leyland Factoring Estimates for a 1024-Bit RSA Modulus. in *Advances in Cryptology - Asiacrypt, LNCS 2894*, pages 55–74, 2003.
21. T. Kleinjung Cofactorisation strategies for the number field sieve and an estimate for the sieving step for factoring 1024 bit integers. `http://www.hyperelliptic.org/tanja/SHARCS/talks06/thorsten.pdf`, 2005.

ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France.
E-mail address: `shih.bai@gmail.com`

INRIA Nancy - Grand Est, Villers-lès-Nancy, France.
E-mail address: `cyril.bouvier@inria.fr`

INRIA Nancy - Grand Est, Villers-lès-Nancy, France.
E-mail address: `alexander.kruppa@inria.fr`

INRIA Nancy - Grand Est, Villers-lès-Nancy, France.
E-mail address: `paul.zimmermann@inria.fr`