INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Computation of Irregular Primes up to Eight Million (Preliminary Report)

M.A. Shokrollahi

TR-96-002

January 1996

Abstract

We report on a joint project with Joe Buhler, Richard Crandall, Reijo Ernvall, and Tauno Metsänkylä dealing with the computation of irregular primes and cyclotomic invariants for primes between four and eight million. This extends previous computations of Buhler et al. [4]. Our computation of the irregular primes is based on a new approach which has originated in the study of Stickelberger codes [13]. It reduces the problem to that of finding zeros of a polynomial over $\mathbb{F}_p$ of degree $< (p-1)/2$ among the quadratic residues. Use of fast polynomial gcd-algorithms gives an $O(p \log^2 p \log\log p)$-algorithm for this task. By employing the Schönhage-Strassen algorithm for fast integer multiplication combined with a version of fast multiple evaluation of polynomials we design an algorithm with running time $O(p \log p \log\log p)$. This algorithm is particularly efficient when run on primes p for which $p-1$ has small prime factors. We also give some improvements on the previous implementations for computing the cyclotomic invariants of a prime.

1 The Problem

In what follows p always denotes an odd prime.

The Bernoulli numbers B_t are rational numbers defined by the identity

$$\frac{x}{e^x - 1} = \sum_{t=0}^{\infty} B_t \frac{x^t}{t!}.$$

The odd Bernoulli numbers are zero except for B_1. A pair $(p, 2t)$, $0 < t < (p-1)/2$ is called *irregular*, iff p divides B_{2t}. The number of irregular pairs for p is called the *index of irregularity* of p, and is denoted by $i(p)$; p is called *regular* if $i(p) = 0$, and is called *irregular* otherwise. In this article, we will develop new methods for computing all the irregular pairs for given p.

The irregular pairs for p can be used to compute several other interesting number theoretic data, as described in Section 3.

The irregular pairs were computed for primes less than 125000 by Wagstaff [17]. This paper also contains a nice account on the history of these computations prior to its appearance. Wagstaff's tables were extended by Tanner and Wagstaff [15] to primes below 150000. Both these approaches were quadratic, i.e., their running time was proportional to p^2. Buhler, Crandall, and Sompolski [3] were the first to invent an $O(p^{1+\varepsilon})$-algorithm for this task and used their method to extend the computations of irregular pairs to one million. Their elegant approach is based on the inversion of the power series $(e^x - 1)/x = \sum_{i=1}^{\infty} x^{k-1}/k!$ modulo x^{p-1} over $\mathbb{F}_p$. This method, combined with further memory saving strategies, was taken up again by Buhler, Crandall, Ernvall, and Metsänkylä [4] to extend previous computations to four million.

In this article, we report on a new approach to compute irregular pairs for primes below eight million. The resulting method is asymptotically comparable to Buhler et al.'s approach, but allows for great memory and running time savings if $p-1$ has only "small" prime divisors. This will be explained in detail in the next section.

2 An Algorithm

In the following p denotes an odd prime, w is a primitive root mod p regarded as a positive integer less than p, and c is an integer satisfying $1 \le c \le p-1$. $\mathbb{Q}$ is the field of rational numbers, $\mathbb{Z}_p$ denotes the ring of p-adic integers, and $\mathbb{F}_p$ is the finite field with p elements. Let

$$h_c(x) := \sum_{j=0}^{p-2} \left\lfloor \frac{c(w^{-j} \bmod p)}{p} \right\rfloor x^j \in \mathbb{F}_p[x].$$

This polynomial arises as a generator polynomial for the Stickelberger code, see [13]. Our first aim is to prove the following.

Theorem 1. *For all* $1 < k \le p-2$ *we have*

$$h_c(w^k) \equiv (c - c^k)\frac{B_{p-k}}{p-k} \bmod p.$$

Proof. Our proof is a slight modification of the argument given in [13]. The elements of the galois group G of the pth cyclotomic field over $\mathbb{Q}$ are the $\sigma_c \colon \zeta \mapsto \zeta^c$, where $\zeta = e^{2\pi i/p}$. The group ring $\mathbb{F}_p[G]$ is isomorphic to $\mathbb{F}_p[x]/(x^{p-1}-1)$ via the morphism φ of $\mathbb{F}_q$-algebras sending σ_w to $x \bmod (x^{p-1}-1)$.

Let $\theta := \frac{1}{p}\sum_d d\sigma_d^{-1} \in \mathbb{Q}[G]$ be the Stickelberger element. Then

$$\begin{aligned}(c-\sigma_c)\theta &= \sum_{d=1}^{p-1}\left(\frac{cd}{p} - \frac{(cd) \bmod p}{p}\right)\sigma_d^{-1} \\ &= \sum_{d=1}^{p-1}\left\lfloor \frac{cd}{p}\right\rfloor \sigma_d^{-1} \\ &= \sum_{j=0}^{p-2}\left\lfloor \frac{c(w^{-j} \bmod p)}{p}\right\rfloor \sigma_w^{j}.\end{aligned}$$

We thus deduce that $(c-\sigma_c)\theta \bmod p$ is a well-defined element of $\mathbb{F}_p[G]$ and

$$\varphi((c-\sigma_c)\theta \bmod p) = h_c \bmod (x^{p-1}-1).$$

Let ω be the Teichmüller character of G, i.e, $\omega(\sigma_c) \equiv c \bmod p$, and consider the central primitive idempotents $\varepsilon_{\omega^k} := \frac{1}{p-1}\sum_{c=1}^{p-1}\omega^k(\sigma_c)\sigma_c^{-1}$ of $\mathbb{Z}_p[G]$ corresponding to ω^k for $0 \le k \le p-2$. Note that

$$\varepsilon_{\omega^k}\sigma_c = \omega^k(\sigma_c)\varepsilon_{\omega^k}. \tag{1}$$

We deduce that

$$\varepsilon_{\omega^k}(c-\sigma_c)\theta = (c-\omega^k(\sigma_c))B_{1,\omega^{-k}}\varepsilon_{\omega^k},$$

where $B_{1,\omega^{-k}} := \frac{1}{p}\sum_{c=1}^{p-1} c\omega^{-k}(\sigma_c)$ is the *generalized Bernoulli number* corresponding to ω^{-k}. We remark that $B_{1,\omega^{-k}} \in \mathbb{Z}_p$ for $i \neq 1$, and $B_{1,\omega^{-k}} = 0$ if k is nonzero and even. (Cf. [18, p. 31]). We further deduce from (1) that for all $\gamma \in \mathbb{Z}_p[G]$ and all $k \neq 1$ we have

$$\varphi(\varepsilon_{\omega^k}\gamma \bmod p) \equiv \varphi(\gamma \bmod p)(w^k)\varphi(\varepsilon_{\omega^k} \bmod p) \bmod p,$$

where $\varphi(\gamma \bmod p)(w^k) \in \mathbb{Z}_p$ is the value of $\varphi(\gamma \bmod p)$ (regarded as a polynomial over $\mathbb{Z}_p$) at w^k. (We have to exclude $k=1$ since $B_{1,\omega^{-1}} \notin \mathbb{Z}_p$.) Altogether this gives

$$h_c(w^k) \equiv (c - c^k)B_{1,\omega^{-k}}$$

for $0 \le k \le p-2$, $k \neq 1$. Now we use the congruence

$$B_{1,\omega^n} \equiv \frac{B_{n+1}}{n+1} \bmod p$$

holding for all odd n satisfying $n \not\equiv -1 \bmod p-1$, see [18, Corollary 5.15]. This gives the assertion of the theorem for odd k with $1 < k \le p-2$. For even k in this range the congruence is trivially valid since both sides vanish mod p. $\square$

The theorem says that $(p, 2t)$ is a irregular pair iff $h_w(w^{p-2t}) = 0$. For computing the irregular pairs we thus need to compute the zeros of h_w among the quadratic nonresidues of $\mathbb{F}_p$. As all the quadratic residues except possibly 1 are zeros of h_c, we may as well work with the polynomial η defined by $\eta(x) = (x-1)h_w(x)/(x^{(p-1)/2}-1)$. The problem we are concerned with is thus that of finding the zeros of the polynomial η of degree $< (p-1)/2$ among the quadratic nonresidues of $\mathbb{F}_p$.

The first idea to solve this problem would probably be to perform a polynomial gcd, followed by a factorization. Even neglecting the cost of the factorization step, this algorithm would be of order $O(p \log^2 p \log\log p)$. We are interested in a faster algorithm, i.e., one of order $O(p \log p \log\log p)$.

Following a suggestion of Schönhage [11], we will first transform this problem to an instance of integer multiplication. For reasons to become clear later, we will consider a more general setup:

Problem 2. *Given a factor d of $p-1$ such that $(p-1)/d$ is even, a polynomial f over $\mathbb{F}_p$ of degree less than d, and a coset C of the cyclic subgroup of $\mathbb{F}_p^\times$ of order d, find all zeros of f in C.*

Let $f(x) = \sum_{i<d} f_i x^i$, and let u be a generator of the subgroup of order d of $\mathbb{F}_p^\times$. Further, let α be a representative of the coset C, i.e., $C = \{\alpha u^i \mid 0 \le i < d\}$. In a first step we multiply the f_i with α^i to obtain $g(x) = \sum_{i<d} f_i \alpha^i x^i =: \sum_{i<d} g_i x^i$. We thus need to compute zeros of g among the elements u^j, $0 \le j < d$. The standard procedure to solve this problem is to employ a technique known as Bluestein's trick: as $(p-1)/d$ is even, there exists $v \in \mathbb{F}_p$ such that $v^2 = u$. As $g(u^j) = g(v^{2j})$, we obtain for all $0 \le j < d$

$$g(u^j) = \sum_i g_i v^{i^2+j^2-(j-i)^2} = v^{j^2} \sum_i (g_i v^{i^2}) v^{-(j-i)^2}. \tag{2}$$

Let $h_i := g_i v^{i^2}$, and $v_i := v^{-i^2}$. If d is odd, then $v^{d^2} = -1$, hence the above is a negacyclic convolution of the vectors (h_i), and (v_i). If d is even, this is a cyclic convolution of these vectors. In any event, one can obtain $v^{-j^2} g(u^j)$ by multiplying the polynomials $h(x) := \sum_i h_i x^i$ and $v(x) := \sum_i v_i x^i$, and performing a wrap-around (with negative or positive sign, according to whether d is even or odd). We have thus reduced the problem to that of multiplying two polynomials of degree less than d over $\mathbb{F}_p$. We now use Schönhage's technique as presented in [9] to reduce this problem to that of multiplying integers. Let $m := \lceil \log(dp^2) \rceil$. (Here and in the sequel log denotes $\log_2$.) Further, let

$$\left(\sum_{i=0}^{d-1} h_i 2^{mi}\right) \cdot \left(\sum_{i=0}^{d-1} v_i 2^{mi}\right) = \sum_{i=0}^{2d-2} c_i 2^{mi}.$$

As $\sum_{l+j=i} h_l v_j < 2^m$ for all i, we obtain $c_i = \sum_{l+j=i} h_l v_j$, hence $h(x)v(x) = \sum_{i=0}^{2d-2} (c_i \bmod p) x^i$. The running time of this algorithm is clearly dominated by the time needed to multiply two integers of bit-length dm, which, using the Schönhage-Strassen algorithm [12], is $O(dm \log dm \log\log dm)$.

The first version of our algorithm was the implementation of this idea, with $d = (p-1)/2$, C the coset consisting of the quadratic nonresidues, and $f = \eta$. (Timings for this and the following more sophisticated versions on some specific primes are provided by the table at the end of this section.)

A major improvement can be gained by employing an idea related to the multiple evaluation algorithm of Borodin and Moenck [2]. First, note that by multiplying the ith coefficient of η with w^i, we are left with the problem of finding the zeros of the resulting polynomial among the quadratic residues of $\mathbb{F}_p^\times$. By abuse of notation, we denote the resulting polynomial by the same letter η. Let $q_1 \leq q_2 \leq \cdots \leq q_t$ be the prime factors of $N := (p-1)/2$ in ascending order. For any divisor d of N let C_d denote the subgroup of order d of $\mathbb{F}_p^\times$. If we can quickly compute $f := \eta \bmod (x^{q_t} - w^{jq_t})$ for $j = 1, 3, \ldots, (p-1)/q_t - 1$, then we can apply the technique for solving Problem 2 to compute the zeros of f in the coset $w^j C_{q_t}$. To compute f we apply recursion: start with $f_{1,1} := \eta$. Suppose that we have computed $f_{i,j} = \eta \bmod (x^{d_i} - w^{jd_i})$, where $d_i = q_i \cdots q_t$. From this we compute for ℓ with $\ell < j + (p-1)/d_{i+1}$ and $\ell \equiv j \bmod (p-1)/d_i$ the q_i polynomials $f_{i+1,\ell} := f_{i,j} \bmod (x^{d_{i+1}} - w^{\ell d_{i+1}})$. The computation time for all the $f_{i+1,\ell}$ from all the $f_{i,j}$ is linear in N, and can be done efficiently, as we are computing modulo binomials. Roughly, the whole running time for this version is dominated by the maximum of tN (the time to compute all the polynomials $\eta \bmod (x^{q_t} - w^{jq_t})$) and N/q_t times the time required to multiply two $q_t m$-bit integers, where, as above, $m = \lceil \log(p^2 q_t) \rceil$. For all primes p for which $p-1$ consists of many prime factors, this version improves significantly over the last one, see the table at the end of this section. One reason for this is that the Schönhage-Strassen algorithm for multiplication of integers is not a linear time algorithm, and that we can efficiently break down the problem into many smaller pieces.

Further savings can be achieved by noting that whenever we apply the solution of Problem 2 to compute the zeros of the polynomial f in a certain coset of the subgroup of order q_t of $\mathbb{F}_p^\times$, we are actually computing the polynomial product of $h(x)$ and $v(x)$. The main point is now that v is fixed for all the cosets. So we are actually dealing with the problem of multiplying one integer with several other integers. This problem can be solved efficiently in the following way: we use the Schönhage-Strassen algorithm for multiplying integers. In a first step we generate v, and its Fourier transform and store it. Then, for each h encountered, we compute its transform, multiply it with the transform of v, and transform the product back. This reduces the number of Fourier transforms per polynomial multiplication from three to two. Another improvement can be gained using an idea of D. Reischert [8]: If p is not a Fermat prime, then the largest prime factor q_t of $p-1$ is odd, hence we have to perform a negacyclic convolution of $h(x)$ and $v(x)$. Translated into integer multiplication, this means that we are performing multiplication modulo $2^m + 1$ for some m. If m is such that $32m = 2^k l < (2k-1)2^{2k}$, then one can perform the Schönhage-Strassen multiplication algorithm to compute this integer product. (See [10, p. 32].) The advantage over the method which multiplies h and v as integers and reduces the product mod $x^{q_t} + 1$ is that the length of the numbers to be multiplied is smaller. (We do not need zero-padding in this version.) However, since $32m$ might contain a small power of 2 (usually 2^6), the new method maybe slower than the old one for some values of m. In our implementation the new method computes the smallest l' such that $q_t(p-1)^2/2 < 2^{32l'}$ and performs a Schönhage-Strassen multiplication modulo $2^{32l'} + 1$, while the old method computes the smallest l such that $q_t(p-1)^2/2 < 2^l$ and performs multiplication modulo $2^m + 1$, where $m = \lceil 2lq_t/32 \rceil$. The final decision whether to use the old or the new method was made according to whether $\lceil l/32 \rceil q_t$ was larger than $\lceil 2lq_t/32 \rceil$ or not.

We have implemented all the three versions of our algorithm described above. The lion's share of the implementations has been done in TP-code. TP which is an invention of A. Schönhage, simulates a multi tape Turing machine. It is a software implementation of a Turing Processor, which can be programmed via *TPAL*, the Turing Processor Assembly Language. Currently, there exists a substantial collection of algorithms written in *TPAL*, including the classical routines for computing with integers and many of the asymptotically fast algorithms for this domain. These clean and efficient implementations made TP the natural choice to implement our algorithm in. For more information on this software and how to obtain it via ftp, the reader is referred to the TP-book [10].

We finish this section by providing some timings for the three versions of our algorithm run on different primes. The computations were done on a SPARC-10 with 40 MHZ. The TP-code was compiled with E. Vetter's tpc, which is a portable *TPAL*-compiler based on the GNU C-compiler. (See [10, Chapter 3].) All the timings are in seconds. They were measured with the UNIX time-command.

p	Factors of $(p-1)/2$	Version 1	Version 2	Version 3
8731	(3,3,5,97)	1.43	0.44	0.42
17467	(3,41,71)	2.82	1.47	1.39
29917	(2,3,3,3,277)	5.22	2.07	1.85
34939	(3,3,3,647)	6.23	3.51	2.58
59863	(3,11,907)	11.88	6.67	4.68
119737	(2,2,3,3,1663)	27.59	14.01	13.99
139801	(2,2,3,5,5,233)	31.47	13.86	8.35
239489	(2,2,2,2,2,2,1871)	62.67	32.12	28.15
279607	(3,46601)	67.35	64.16	83.94
478991	(5,19,2521)	130.75	62.49	55.30
559217	(2,2,2,7,4993)	171.45	100.20	69.41
957991	(3,5,11,2903)	285.38	155.06	102.44
1118437	(2,3,11,37,229)	410.57	106.10	78.93
3238481	(2,2,2,5,7,5783)	–	646.03	366.40
3988961	(2,2,2,2,5,107,233)	–	479.64	382.56
6000149	(2,7,7,11,11,11,23)	–	1357.05	422.57
6000191	(5,7,85717)	–	–	1350.58
12000097	(2,2,2,2,3,3,3,17,19,43)	–	–	1152.30
16000129	(2,2,2,2,2,2,3,3,17,19,43)	–	–	1560.22
20001301	(2,3,5,5,11,11,19,29)	–	–	1481.49

3 Applications

Once the irregular pairs for a prime p are computed, one can use this to compute other arithmetic data of the cyclotomic field K_0 generated over $\mathbb{Q}$ by a primitive pth root of unity. In this section we will report on two such applications, which, up to now have always accompanied the computations of the irregular pairs, see [4, 6, 7, 15, 17].

For $n \geq 1$ let K_n denote the cyclotomic field of p^{n+1}st roots of unity, and let h_n and A_n be the class number and p-class group of K_n, respectively. It is well known that

$$h_n = h_n^+ h_n^-, \quad A_n = A_n^+ \oplus A_n^-,$$

where h_n^+ and A_n^+ are the class number and the p-class group of the maximal real subfield of K_n. A conjecture of Vandiver states that p does not divide h_n^+. Using the irregular pairs, we can verify this conjecture for p in the following way: let (p,t) be an irregular pair, L be the smallest prime congruent to 1 modulo p, $m := (L-1)/p$, and

$$q = \prod_{b=1}^{(p-1)/2} (s^b - s^{-b})^{b^{p-1-t}} \bmod L,$$

where s is a square root of a pth root of 1 modulo L. If $q^m \not\equiv 1 \bmod L$ for all t for a given p, then Vandiver's conjecture is true for p. (See Washington's book [18] for details.)

The next application is concerned with the problem of computing the so-called cyclotomic invariants. By Iwasawa's general result and the theorem of Ferrero and Washington we have

$$\mathrm{ord}_p(h_n) = \lambda_p n + \nu_p, \quad \mathrm{ord}_p(h_n^-) = \lambda_p^- n + \nu_p^-,$$

for all n large enough, say $n \geq n_p$, where $\lambda_p, \nu_p, \lambda_p^-, \nu_p^-$ are integers (λ_p, λ_p^- nonnegative) independent of n. Here and in the following $\mathrm{ord}_p(a)$ stands for the exponent of p in the canonical decomposition of a. Suppose that

$$A_n^- \simeq (\mathbb{Z}/p^{n+1}\mathbb{Z})^{i(p)}, \quad (n = 0, 1, \ldots), \tag{3}$$

$$\mathrm{ord}_p(h_0^-) = \mathrm{ord}_p(B_2 B_4 \cdots B_{p-3}). \tag{4}$$

Then, if Vandiver's conjecture is true for p, we obtain

$$\lambda_p = \lambda_p^- = \nu_p = \nu_p^- = i(p), \quad \text{minimal } n_p = 0.$$

(We refer to [6] and the references therein.) Ernvall and Metsänkylä develop in [6] a computational method to decide whether (3) and (4) are true: for an integer a prime to p let q_a denote the Fermat quotient of a, i.e.,

$$q_a = \frac{a^{p-1} - 1}{p} \bmod p, \quad 0 \leq q_a < p.$$

Let (p,t) be an irregular pair, $N = (p-1)/2$, and put

$$S_1 := \sum_{a=1}^{N} a^{t-1} q_a, \quad S_3 = \sum_{a=1}^{N} a^{t-1}.$$

If $S_1 \not\equiv 0 \bmod p$, $S_3 \neq (1-t)pS_1 \bmod p^2$, and $S_3 \not\equiv 0 \bmod p^2$, then (3) and (4) hold for p. Ernvall and Metsänkylä have suggested the following procedure for computing these sums: note first that we have the following congruences:

$$q_{2a} \equiv q_2 + q_a \bmod p, \qquad q_{p-2a} \equiv q_2 + q_a + (2a)^{p-2} \bmod p \quad \text{if } (p-1)/2 < 2a < p.$$

We can thus compute the sums S_1 and S_3 in the following way. The q_{2a} are computed in cycles, passing from q_a to q_{2a} or to q_{p-2a} according to whether $2a < N$ or not. They are stored in an array consisting of N 32-bit integers, where the ath entry corresponds to q_a. (We are in the range $p < 2^{32}$.) When a new q_a is computed, it is first checked whether the corresponding entry is zero. If not, this means that the computation over the current cycle is finished and one can go on to the next a for which the corresponding entry is zero. Along the way, the computed values of q_a are multiplied by a^{t-1} for all t such that (p, t) is irregular. All the computations are carried out modulo p^2. (Although S_1 is only needed modulo p, we need to perform computations modulo p^2 for computing the q_a and hence S_1.) At the end of these computations, one eventually obtains the desired values for S_1 and S_3. The same implementation has been used to compute the cyclotomic invariants up to four million [4].

It is clear that this method requires at least $4N$ bytes of memory for storing the quotients q_a. Our slightly modified method reduces the memory requirement by a factor of $1/32$, at the expense of some overhead in the computation. The idea is as follows: ignoring diophantine constraints, we allocate an array of integers of length $N/32$, call it `fq`, and regard it as an array of N bits. This array is initialized with zeros at the beginning. Whenever we compute some q_a, we check whether the ath bit of `fq` is set to one. If not, we set this bit to one, and add $a^{t-1}q_a \bmod p$ to the previous value of S_1, and $a^{t-1} \bmod p^2$ to the previous value of S_3. If the ath bit of `fq` is one, we know that the computation over one orbit is finished, and we can scan in 32-bit steps over the array `fq` for an a whose orbit has not been involved in the computation yet. The computational overhead we encounter here is the reading and the setting of bit-values in `fq` for each a. Both these operations can be performed efficiently by ORing or ANDing with appropriate powers of two ($2^0 - 2^{31}$), which are computed in advance.

4 Computations

Our computations were carried out on spare cycles of a cluster of 42 SPARC-10 and SPARC-20 machines with altogether 56 processors. These machines were kindly provided by the department of computer science, Universität Bonn, Abteilung II and III, and the Gesellschaft für Mathematik und Datenverarbeitung. They varied quite a lot with respect to their RAM, their virtual memory, and their number of processors. No machine was equipped with less than 40 MB of RAM. In order to keep the load of the machines used at a minimum, on all but one machine with two processors we only ran our program on primes which consumed no more than 40 MB of memory during the course of computations. For each prime we estimated in advance the amount of memory used via the empirical formula

$$\text{Amount of memory used} \sim (650 * \text{Largest prime factor of } p - 1)\text{bytes}.$$

A secondary program generated all primes between four and eight million, estimated the amount of memory used, and passed the data to a resource management program (RMP) written by E. Vetter [16]. The RMP passed the prime to the first machine available which had the amount of RAM necessary to compute the irregular pairs corresponding to the prime in question. Furthermore, the RMP also allowed the use of specific time windows for computations on different machines (usually during the night and at weekends). All information regarding the profile of the machines were kept in a file, which the RMP always consulted before passing a prime to an available machine. In that way, we could quickly react to possible changes of the computing environment.

The computations started in April 1995, and stopped in December 1995. During this period, we could compute the indices of 157,049 primes in the range between four and eight million. (There are 256,631 primes in this range.) The remaining primes will be computed via a different implementation by Joe Buhler and Richaed Crandall using the computing facilities of the NeXT-corporation in California. We will use the two implementations to randomly cross-check the results obtained by us and by Buhler-Crandall's implementation. Large-scale computations are vulnerable to all sorts of errors, so besides the cross-check facility, it is good to have some sort of checksum identity to check the results. Buhler et al. [3, 4] use the identity $\sum_{n=0}^{p-3} 2^n(n+1)B_n \equiv -4 \bmod p$. In our case we used the following identity: if f is a polynomial over $\mathbb{F}_p$ of degree less than $N = (p-1)/2$, then $\sum_t f(t) = Nf(0)$, where t runs over the quadratic residues of $\mathbb{F}_p$.

Among the primes investigated by our program, we found two of index seven: 5,216,111 and 5,620,861. Previously the only prime of index seven known was 3,238,481 [4]. As is noted by Buhler et al. [4], conjectures regarding the distribution of indices leads one to guess that the smallest prime of index seven should be about 16,500,000, so the fact that there are at least three primes below this number is surprising.

5 An Open Problem

It is well-known that there are infinitely many irregular primes. It is not known whether there are infinitely many regular primes. Nevertheless, if the numerators of the Bernoulli numbers are uniformly random modulo odd primes, then the index of irregularity should satisfy a Poisson distribution with mean 1/2. As a result, the density of the irregular primes should be $1-e^{-1/2}$, which is roughly 39.3%. In other words, about 60% of the primes should be regular. For our computations this means that in almost 60% of the cases we perform a very complicated computation to obtain one bit of information.

The question now is the following: Is there a possibility to quickly decide whether a given prime is regular or not? Here, we mean by a "quick" algorithm one that runs in time $O(p)$, preferably with small constants. According to our description of the indices of irregularity of a prime in terms of the zeros of the polynomial η, we may as well ask whether there is a possibility to quickly decide for a given polynomial over $\mathbb{F}_p$ whether it has a zero among the quadratic nonresidues (or quadratic residues). We can also state this problem in a randomized setting: design an algorithm which on input $f \in \mathbb{F}_p[x]$ outputs *YES* or *NO* according to whether f has a zero among the quadratic residues of $\mathbb{F}_p$, with error probability 0 if the answer is *NO*.

One possible strategy to solve the above problem would be to compute the product $\prod_t f(t)$ in time $O(p)$, where t ranges over the quadratic residues. Below we shall derive an $O(p \log p)$ lower bound for this problem in the model of straight-line programs. This result may suggest that it is difficult to find a linear time algorithm for computing the above product. (For a thorough discussion of the model of straight-line programs and the lower bound techniques used in the sequel we refer the reader to the forthcoming book [5].)

Let f be a polynomial of degree n with indeterminate coefficients over a field k and denote the field generated over k by the coefficients of f by K. Let $x_0, \ldots, x_{n-1}$ be indeterminates over K, and $F := f(x_0)f(x_1)\cdots f(x_{n-1})$. We are interested in a lower bound for the non scalar complexity $L(F)$ of F over K. Let ∂_i denote the derivation morphism $\partial/\partial x_i$ of $K(x_0, \ldots, x_{n-1})$. By the Baur-Strassen derivative inequality [1] we have

$$3L(F) \geq L(F, \partial_0 F, \ldots, \partial_{n-1} F).$$

Note that $\partial_i F = Ff'(x_i)/f(x_i)$. Hence, the right-hand side of the above inequality is at least $L(f'(x_0)/f(x_0), \ldots, f'(x_{n-1})/f(x_{n-1}))$, which itself is greater than or equal to $L(f'(x_0), \ldots, f'(x_{n-1})) - 2n$. A simple application of Strassen's Degree Bound [14] gives a lower bound of $n \log(n-1) - 2n$ for the latter.

6 Acknowledegements

The use of fast integer multiplication to find the zeros of a polynomial over a finite prime field has been a suggestion of Prof. A. Schönhage whose help is hereby greatly acknowledged. Furthermore, it is a pleasure to thank him for introducing to me the world of TP, for the many fruitful discussions on TP and other topics, and for his support to get computing time on the machines of Abteilung II, Department of Computer Science, Universität Bonn.

I also have used machines of Abteilung III of the Department of Computer Science of Universität Bonn, and of the Gesellschaft für Mathematik und Datenverarbeitung (GMD). I would like to thank Prof. A.B. Cremers (Department of Computer Science, Bonn), and Prof. Th. Lengauer (GMD) for providing me with the necessary hardware and for their interest in this work.

A small part of the computations were done on a machine of the University of Turku. I would like to thank T. Metsänkylä and the University of Turku for providing me with computation time on this machine.

Special thanks go to J. Buhler, R. Crandall, and T. Metsänkylä for sharing with me their experience with Bernoulli computations and for providing me with their earlier programs, and to J. Buhler for comments on an earlier draft of this report.

Whenever I needed any TP- or Unix-specific suggestion, E. Vetter was there to help, and he didn't stop until the problem was solved. The Resource Management Program was entirely written by him. Without his continuous support, this project would have probably failed.

Almost all the sophisticated parts of the TP-code have been written by D. Reischert. It is a pleasure to thank him for providing me with his excellent software.

Finally, many thanks go to several system administrators of the machines I worked on for their patience with my project and for their support.

References

[1] W. Baur and V. Strassen: The complexity of partial derivatives. *Theoret. Comp. Sci.*, **22**, 317–330, 1983.

[2] A. Borodin and R. Moenck: Fast modular transforms. *J. Comp. Syst. Sci.,* **8**, 366–386, 1974.

[3] J. P. Buhler, R. E. Crandall, and R. W. Sompolski: Irregular primes to one million. *Math. Comp.*, **59**, 717–722, (1992).

[4] J. P. Buhler, R. E. Crandall, R. Ernvall, and T. Metsänkylä: Irregular primes and cyclotomic invariants to four million. *Math. Comp.*, **61**, 151–153, (1993).

[5] P. Bürgisser, M. Clausen, and M. A. Shokrollahi: *Algebraic Complexity Theory.* To appear in *Springer Verlag*, Heidelberg.

[6] R. Ernvall and T. Metsänkylä: Cyclotomic invariants for primes between 125000 and 150000. *Math. Comp.*, **56**, 851–858, 1991.

[7] R. Ernvall and T. Metsänkylä: Cyclotomic invariants for primes to one million. *Math. Comp.*, **59**, 249–250, 1992.

[8] D. Reischert: Private Communication.

[9] A. Schönhage: Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients. In *Computer Algebra EUROCAM '82 (Marseille 1982)*, ed. J. Calmet. *Lect. Notes Comp. Sci.*, **144**, 3–15.

[10] A. Schönhage, A. F. W. Grotefeld, and E. Vetter: *Fast Algorithms. A Multitape Turing Machine Implementation.* B.I. Wissenschaftsverlag, Mannheim, 1994.

[11] A. Schönhage: Private communication.

[12] A. Schönhage: Schnelle Multiplikation großer Zahlen. *Computing*, **7**, 281–292, 1971.

[13] M. A. Shokrollahi: Stickelberger Codes. *To appear in Designs, Codes, and Cryptography.*

[14] V. Strassen: Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten. *Numer. Math.*, **20**, 238–251, 1973.

[15] J. W. Tanner and S. S. Wagstaff: New congruences for the Bernoulli numbers. *Math. Comp.*, **48**, 341–350, (1987).

[16] E. Vetter: Private communication.

[17] S. S. Wagstaff: The irregular primes to 125000. *Math. Comp.*, **32**, 583–591, (1978).

[18] L. Washington: *Introduction to Cyclotomic Fields.* Springer Verlag, New York, 1982.