Computing Discrete Logarithms with Quadratic Number Rings

Damian Weber

Institut für Techno– und Wirtschaftsmathematik
Erwin–Schrödinger–Str. 49
D–67663 Kaiserslautern
Germany
e-mail: weber@itwm.uni-kl.de

Abstract

At present, there are two competing index calculus variants for computing discrete logarithms in $(\mathbb{Z}/p\mathbb{Z})^*$ in practice. The purpose of this paper is to summarize the recent practical experience with a generalized implementation covering both a variant of the Number Field Sieve and the Gaussian integer method. By this implementation we set a record with p consisting of 85 decimal digits. With regard to computational results, including the running time, we provide a comparison of the two methods for this value of p.

Keywords: Discrete Logarithms, Number Field Sieve, Index Calculus

1 Introduction

Let $(G, \cdot)$ be a finite group and $a, b \in G$. If there exists $x \in \mathbb{Z}$, $x \geq 0$, such that

$$a^x = b, \tag{1}$$

we call the minimal x satisfying (1) the *discrete logarithm* of b to the base a.

In this article we consider the cyclic group $G = (\mathbb{Z}/p\mathbb{Z})^*$ of the finite prime field with characteristic p. This is the group the Number Field Sieve discrete log algorithm has been designed for [6]. For ease of use we call it NFS–DL. At ANTS–II (1996) we presented a computation involving a 65–digit p [13]. Our latest endeavours resulted in a computation with a 85–decimal–digit $p = 2q + 1$ where q is prime.

As with factoring we have two different algorithms for the most difficult problem instances. On the one hand we have the Gaussian integer algorithm COS [1], a special case of the Number Field Sieve algorithm [12] with conjectured running time $L_p[\frac{1}{2}, 1]$, on the other hand the NFS-DL algorithm [6, 11] with conjectured running time $L_p[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}}]$, where

$$L_p[s, c] := \exp((c + o(1))\,(\log p)^s (\log\log p)^{1-s}).$$

It remains the question: when does the NFS–DL beat COS?

This paper provides a partial answer: NFS–DL is slower as long as p does not consist of more than 85 decimal digits.

2 Sketch of the NFS Algorithm

We start by providing a rough outline how our implementation computes discrete logarithms. In the subsequent sections we will focus our attention to steps 1, 2 and 5. In step 1, a new sieving method is employed, which reduces the size of the right hand side of the discrete logarithm problem (1). In step 2, we were able to use Montgomery's method of finding two quadratic polynomials with small coefficients, which produces two suitable (quadratic) number rings [9]. Different from the factoring case, however, a few additional relations have to be found when applying this variation (step 5). For an explicit description of step 6, we refer the reader to [13].

1. reduce the original problem (1) to congruences

$$a^x \equiv s \bmod p,$$

   $s \in S$ where S is a set of "sufficiently" small natural numbers
2. choose two polynomials $g_1(X), g_2(X) \in \mathbb{Z}[X]$ of degree n_1, n_2 respectively with common root m mod p;
   for $j = 1, 2$:
   - let $h_j \in \mathbb{Z}$ be the coefficient of X^{n_j} in polynomial g_j,
   - let $\alpha_j \in \mathbb{C}$ be a root of g_j
   - let $\mathcal{O}_j \supset Z[h_j\alpha_j]$ be the ring of integers of $K_j := \mathbb{Q}(\alpha_j)$

   each $s \in S$ must split into first degree prime ideals in one of the $\mathcal{O}_j$ (*)
3. choose factor base bounds $B_j \in \mathbb{Z}$ and factor bases
   $F_j = \{\text{first degree prime ideals of } \mathcal{O}_j \text{ with norm } \leq B_j\} \cup \{h_j\mathcal{O}_j\}$
   and large prime bounds $L_j \in \mathbb{Z}$
4. find set of pairs $C := \{(c, d)\} \subset \mathbb{Z} \times \mathbb{Z}$ with
   $h_1 \cdot (c + d\alpha_1)$ smooth over F_1
   $h_2 \cdot (c + d\alpha_2)$ smooth over F_2 by sieving, with $|C| > |F_1| + |F_2|$
5. for each $s \in S$ choose $j := j_s \in \{1, 2\}$, such that s satisfies condition (*) in $\mathcal{O}_j$; then find special relations:
   $h_1 \cdot (c + d\alpha_j)/\mathfrak{p}_s$ smooth over F_1
   $h_2 \cdot (c + d\alpha_{3-j})$ smooth over F_2
   for each prime ideal $\mathfrak{p}_s \subset \mathcal{O}_j$ lying above s
6. for every "sufficiently large" prime divisor q dividing $p - 1$:
   - construct a sparse matrix A by processing the data found in steps 4 and 5
   - solve $Ax \equiv 0 \bmod q$

   For smaller prime divisors q, apply the Pohlig–Hellman method [10]. Without any difficulties, this method may be applied to $q \leq 10^{16}$ [13].

3 The Reduction Step

This section is devoted to step 1 of the survey at the end of the previous section. We are interested in reducing the original task of solving

$$a^x \equiv b \bmod p$$

to several tasks

$$a^{x_i} \equiv s_i \bmod p, \quad s_i \in \mathbb{Z}, \quad 1 \leq i \leq k,$$

where $s_i \leq B$ for some bound B. In practice, $B \approx 10^{10}$ is a reasonable size.

Such a reduction is necessary because in the NFS step 5 the simultaneous smoothness of the terms (let s run through the s_i)

$$(c + d\alpha_j)/\mathfrak{p}_s \text{ and } (c + d\alpha_{3-j}), \quad \text{with } \mathfrak{p}_s \supset s\mathcal{O}_j \tag{2}$$

is required. The difficulty of finding a relation of type (2), however, raises with the size of s: let the prime ideal $\mathfrak{p}_s$ be generated by $(s, \alpha_j - r)$ over $\mathcal{O}_j$, $r \in \mathbb{Z}$. Then $\mathfrak{p}_s$ divides $(c + d\alpha_j)$ if and only if $-c/d \equiv r \bmod s$. So we expect either c or d to be of size s. The subject of this section is a new sieving method which can be used to minimize the maximal value of s for a given DL problem.

Our method resembles the computation of individual logarithms with the residue list sieve (see [1]). The difference lies in allowing simultaneous sieving of two polynomials instead of performing two separate sieving steps.
To begin with, fix some smoothness bound B; usually this will be one of the large prime bounds L_j.

For $b \in \mathbb{Z}$, we find a representation $b \equiv t_1/t_2 \bmod p$, $|t_1|$, $|t_2| \leq \sqrt{p}$, as follows: compute the partial quotients p_l/q_l of the continued fraction of b/p until $q_l < \sqrt{p} < q_{l+1}$. We know (see, for example, [7, Th. 10.2.4])

$$\left| \frac{b}{p} - \frac{p_l}{q_l} \right| \leq \frac{1}{q_l q_{l+1}}.$$

Set $t_2 := q_l$, $t_1 := q_l b - p_l p$. We have $t_2 < \sqrt{p}$ by construction and $|t_1| < \sqrt{p}$ because of

$$|p_l p - q_l b| = |pq_l| \cdot \left| \frac{b}{p} - \frac{p_l}{q_l} \right| \leq \frac{pq_l}{q_l q_{l+1}} < \sqrt{p}.$$

So we are left with the following problem. Given a number $t \leq \sqrt{p}$, find a B–smooth representation of t. Multiply t by an appropriate power of 2 such that $\sqrt{p}/2 \leq t \leq \sqrt{p}$. Set $t' := \lfloor \frac{p}{t} \rfloor + 1$ and define homogeneous polynomials of degree one as

$$\begin{aligned} f_1(X, Y) &:= X + t'Y \\ f_2(X, Y) &:= tX + (tt' - p)Y. \end{aligned} \tag{3}$$

Search for $(x, y) \in \mathbb{Z}^2$, such that f_1, f_2 are simultaneously B–smooth. This can be achieved by sieving. Having found such a pair we have

$$\begin{aligned} x + t'y &= \prod_{p \le B} p^{e_p} \\ tx + tt'y &\equiv t(x + t'y) \\ &\equiv \prod_{p \le B} p^{e'_p} \bmod p \end{aligned} \tag{4}$$

and therefore

$$t \equiv \frac{\prod_{p \le B} p^{e'_p}}{\prod_{p \le B} p^{e_p}} \bmod p \tag{5}$$

which is a B–smooth representation of $t \bmod p$. In case of x, y being bounded, the size of the numbers tested for smoothness is $O(\sqrt{p})$, as we see from Lemma 1. At the end of this section we give appropriate bounds for x, y which lead to a relation of type (5) with probability close to $1 - 1/e$.

Lemma 1. *Let $C > 0$ be a constant, $x, y \le C$, and f_1, f_2 as in (3). Then*

$$\begin{aligned} f_1(x, y) &\le C(1 + 2\sqrt{p}), \\ f_2(x, y) &\le 2C\sqrt{p}. \end{aligned}$$

Proof.

$$\begin{aligned} &f_1(x, y) \le C(1 + t') \le C(1 + 2\sqrt{p}), \\ &f_2(x, y) = tx + (tt' - p)y \le tx + (t(p/t + 1) - p)y = tx + ty \le 2C\sqrt{p}. \end{aligned}$$

Our experiments show that for p having at most 85 decimal digits, one can expect to find at least one relation which reduces the original DLP modulo p to problems where the right hand side consists of maximal ten decimal digits within acceptable time (table 1). Thus we are left with computing a solution to

$$a^x \equiv s \bmod p,$$

with $s < 10^{10}$.

Table 1. Reduction Step

$\log_{10} p$	max $\lvert x \rvert$	max y	B	# rels	time (s)	time/rel (s)
65	10^6	200	10^9	127	2606	21
75	$2.5 \cdot 10^6$	5040	$1.5 \cdot 10^9$	203	246597	1215
85	$5.0 \cdot 10^6$	735	$2.0 \cdot 10^9$	6	111059	18510

Let us take a look at the row corresponding to the 85–digit number. The numbers t, t' consist of 40 and 45 digits respectively, the maximal value of x is below 10^7, while the maximal value of y is about 10^3. So we split 47–digit and 48–digit numbers into numbers below $B = 2.0 \cdot 10^9$. Sieving the two polynomials f_1, f_2 was done by using a factor base of primes below $3.5 \cdot 10^7$ and allowing one large prime up to $2.0 \cdot 10^9$. So it is required that the second largest prime factor is at most $3.5 \cdot 10^7$. To estimate the running time in order to find a relation satisfying these requirements, it turns out to be useful to have a look at the ρ–function ([8]) which tells us that we can expect this to happen with probability

$$1.90 \cdot 10^{-4} \cdot 3.20 \cdot 10^{-1} \cdot 1.39 \cdot 10^{-4} \cdot 3.11 \cdot 10^{-1} \approx 2.62 \cdot 10^{-7}.$$

Therefore it can be expected that there is one B–smooth value among $3.8 \cdot 10^8$ coprime pairs (x, y). Since there are approximately $3.42 \cdot 10^9$ coprime pairs within the rectangle

$$\{(x, y) \in \mathbb{Z} \times \mathbb{Z} | -5 \cdot 10^6 \leq x \leq 5 \cdot 10^6, 1 \leq y \leq 735\},$$

we expect to find

$$3.42 \cdot 10^9 / 3.8 \cdot 10^8 \approx 9$$

relations producing B–smooth expressions; in fact, we have found six relations. Note that only one of them suffices for satisfying our purposes.

The timings hold for a Sparc 20–167 workstation.

When taking values for B, x and y from table 2 one may expect to find a relation of type (5) with probability close to $1 - 1/e$. In case of not finding any relation, we may proceed with two strategies:

- increase the bounds for x, y, B, or
- generate $b' \equiv a^z \bmod p$ for random z and obtain other values for t_1, t_2.

Table 2. Recommended Parameters for Reduction Sieve

$\log p$	B	x–bound	y–bound
65	10^8	200000	10
75	10^9	350000	100
85	$2 \cdot 10^9$	500000	1000
95	$2 \cdot 10^9$	2000000	10000
105	$2 \cdot 10^9$	30000000	100000

4 Adaptation of the Two–Quadratics-Method

By using our new NFS–DL version with two quadratic polynomials, we solved

$$59^x = 29 \bmod p$$
$$59^y = 53 \bmod p,$$

where

$$p = 3108193808041961141219111205196826101966010119640309197118051941271219700607191207059$$

is a prime of 85 decimal digits with $q := (p-1)/2$ prime.

We computed the two solutions

$$x = 3051032039810976575447505290834805255985233189266022096531322524429784944990676327395 \quad \bmod \ p-1$$

$$y = 1244526127344878448964603523776806303857752904918959081249797500704003169233661409571 \quad \bmod \ p-1.$$

Originally invented by Montgomery [9], the use of two quadratic polynomials in the NFS algorithm has been exploited to achieve impressive factorizations of large integers by Elkenbracht–Huizing [4]. For discrete log problems within the currently solvable range, it turned out to be the preferred method to the standard NFS method considered in [13], see [14]. Considering NFS it seems at first sight that at least one of the number rings should be a principal ideal ring, since computing discrete logarithms of elements requires access to elements of some number ring. Every index calculus algorithm for $\mathbb{Z}/p\mathbb{Z}$ published so far satisfies this need because one of the two number rings is the principal ideal ring $\mathbb{Z}$. With a modification of a few relations, however, we can even compute discrete logarithms if both number rings have class number strictly greater than 1.

To begin with, two quadratic polynomials, which defined the auxiliary number rings, were chosen by Montgomery's method (cf. [5, section 5]) as follows:

$$\begin{aligned} g_1(X,Y) = {} & 12088651913597925810 \cdot X^2 \\ & +905452079113038068089 \cdot XY \\ & +8749043915900881108603Y^2 \\ g_2(X,Y) = {} & 1146890895334804811 \cdot X^2 \\ & +5297984501155169639345 \cdot XY \\ & -3247049136460419754715Y^2. \end{aligned}$$

Set

$$m := 900324066150081045760591947783901178451104941777707468193881618077848960434289779843.$$

Then g_1, g_2 have the common root $(m,1) \bmod p$.

Table 3. Collected Partials

#FB	LP	# relations					mips
		full	single	double	triple	quad	years
70339	$8 \cdot 10^7$	5415	109082	8114437	2563554	5015662	44.5

The factor bases were of size 35346 and 34995 respectively.

The sieving procedure was done on about 100 workstations using their idle time of 44.5 mips years. The following amount of partials was found

Here we pause only to comment on the fact that we do not get # doubles < #triples < quadruples as one would expect (the reader not interested in sieving details may skip this paragraph without loss). This is due to a slight modification of our double large prime variation. In this variation one observes that a non–negligible part of the sieving process is factoring numbers u with $L < u < L^2$ where L is a large prime bound. In case of u being close to L^2 we have two drawbacks: Firstly, factoring $u = l_1 l_2$ is expensive. Secondly, l_1, l_2 do less likely contribute to a full relation, because the bigger the l's are, the smaller is the probability that other relations containing the same l are found. The bottom line is that we want to avoid u's close to L^2. We briefly sketch the way to do this. If a large prime variation gets applied, the sieving bound is changed by the amount of $c \cdot \log L$, where c is the number of large primes allowed. Thus, in the double large prime variation we would normally have c close to 2.0. Our modification is to set $c = 1.3$. A test sieving step with this modification gives satisfying results, see table 4 – we use the usual convention of declaring the large prime relations: for example fppp denotes one large prime for the polynomial g_1 and two large primes for the polynomial g_2. The first observation is that we get more than half of the relations in less than 10 % of the time as denoted in the following table (sieve array $x \in [-600000; 600000]$, $y \in [1, 1000]$). The second observation is that we lose only 10 % of the worthy double large prime relations.

In order to be able to compute the logarithm of two elements with respect to 59, we define $s_1 := 59$, $s_2 := 29$, $s_3 := 53$. By factoring $g_1(X) \bmod s_i$ and verifying that s_i^2 does not divide $\operatorname{disc} g_1$, we know the decomposition of the s_i's into prime ideals $\mathfrak{p}_{11}, \mathfrak{p}_{12}, \mathfrak{p}_{21}, \mathfrak{p}_{22}, \mathfrak{p}_{31}, \mathfrak{p}_{32} \subset \mathcal{O}_1$:

$$\begin{aligned} s_1 \mathcal{O}_1 &= \mathfrak{p}_{11}\mathfrak{p}_{12} \\ s_2 \mathcal{O}_1 &= \mathfrak{p}_{21}\mathfrak{p}_{22} \\ s_3 \mathcal{O}_1 &= \mathfrak{p}_{31}\mathfrak{p}_{32}. \end{aligned}$$

So we have $j_s = 1$ for each s (cf. section 2, step 5). In order to create relations containing the s_i's, we modified three relations (c_i, d_i), $1 \le i \le 3$ according to

$$s_i(c_i + d_i \alpha_1) = \mathfrak{p}_{i1}\mathfrak{p}_{i2} \prod_{\mathfrak{q} \in F_i} \mathfrak{q}^{e_\mathfrak{q}} \quad 1 \le i \le 3. \tag{6}$$

These relations will be used to compute $\log_{s_1} s_2$, $\log_{s_1} s_3$ at the end of this section.

Table 4. Test Sieve for Large Prime Modification

type of relation	$c = 1.9$	cumulated	$c = 1.3$	cumulated
ffff	5	5	5	5
fpff	131		131	
fffp	60	191	60	191
fpfp	1178		1174	
ppff	269		167	
ffpp	190	1637	129	1470
ppfp	2305		1432	
fppp	2967	5272	1839	3271
pppp	6266	6266	2730	2370
total	13371		7667	
time (s)	47686		3434	

In the sequel we consider building the relation matrix. The partial relations resulted in 539 238 full relations by applying the cycle finding algorithm of [16]. As we merely need 70 000 of them, it was advisable to pick the subset of relations which produce not too many entries in our linear system (section 2, step 6). Intuitively, the number of entries (weight) of a full relation correlates with the number of partial relations which contributes to the full relation. This assumption is confirmed by table 5. We learn from this statistics that we actually do not need to build the full relation only to find out that its weight is too high. Instead, we fix a maximal number of partials a full relation should consist of and accept this relation, when fewer partials are contributing to it. The rows of table 5 read as follows. There were 4 322 fulls consisting of 2 partials. The full relation with minimal respectively maximal weight among these consisted of 28 entries respectively 44 entries. The average weight of the fulls consisting of 2 partials was 36.2. In this computation the maximal number of partials per full relation has been set to 19, eventually resulting in 158 841 fulls. Adding 16 235 fulls found by the sieving step, we got a total of 175 046.

In order to actually compute a solution to our DL problem, we need four solutions to eliminate additive characters, which are kept separately, and we need three solutions to produce three congruences of the form

$$s_1^{e_{i1}} s_2^{e_{i2}} s_3^{e_{i3}} \equiv \pm 1 \quad 1 \leq i \leq 3.$$

To be quite confident that the vectors e_i are linearly independent over $\mathbb{Z}/q\mathbb{Z}$, we computed three additional solutions. These considerations lead to computing ten dependencies among the rows of the resulting 175046×70342 linear system modulo q. The linear algebra computation has been split into the following two steps:

Table 5. Correlation of # Partials and Weight

# partials	min weight	max weight	avg weight	# fulls
2	28	44	36.2	4322
3	43	63	52.6	3911
4	57	77	66.3	4241
5	70	93	81.3	5064
6	76	107	94.4	5935
7	96	121	108.7	6963
8	106	136	121.2	8312
9	120	150	135.0	9680
10	129	163	147.3	11046
11	144	177	160.5	12330
12	155	192	172.5	13012
13	169	205	185.5	13354
14	176	214	197.2	13595
15	186	228	209.9	12846
16	199	242	221.4	11825
17	213	258	233.9	9969
18	226	264	245.1	7774
19	232	278	257.4	4662

– a compactification step, which constructed a 54866×54856 system
– the Lanczos algorithm on the parallel PARAGON machine at the KFA in Jülich/ Germany within 64 hours on 64 nodes.

Due to the compactification step, the computing time for the linear algebra step was reduced by more than 30%. The reader, who is interested in the subject of minimizing the running time of a parallel Lanczos implementation by compactification, is referred to [2].

Each solution to the linear system gives exponents $e_{c,d}$ of the elements $c+d\alpha_1$, $c+d\alpha_2$, such that the power products give two q–th powers simultaneously which is the effect of modification (6):

$$\begin{aligned} \textstyle\prod_{i=1}^{3}(s_i(c_i+d_i\alpha_1))^{e_{c_i,d_i}} \cdot \prod_{(c,d)}(c+d\alpha_1)^{e_{c,d}} &= \gamma_1^q \\ \textstyle\prod_{(c,d)}(c+d\alpha_2)^{e_{c,d}} &= \gamma_2^q. \end{aligned} \tag{7}$$

Applying φ_1, φ_2, we get three relations of type (8) from three of type (7).

$$\begin{aligned} s_1^{e_{c_1,d_1}} s_2^{e_{c_2,d_2}} s_3^{e_{c_3,d_3}} \prod_{(c,d)} (c+dm)^{e_{c,d}} &\equiv g_1^q \bmod p, \\ \prod_{(c,d)} (c+dm)^{e_{c,d}} &\equiv g_2^q \bmod p \end{aligned} \tag{8}$$

for some $g_1, g_2 \in \mathbb{Z}$. From the resulting 3×3–system (divide the first congruence of (8) by the second), the logarithms of s_2 and s_3 with respect to s_1 could be computed without difficulty.

5 A New COS Record

In order to get a comparison between NFS and COS, we solved another discrete log problem in the prime field of section 4 by the Gaussian–Integer method. This was achieved by configuring our NFS implementation to meet the requirements of that algorithm.

With the same 85–decimal–digit p as in section 4, we solved the discrete logarithm problem

$$\begin{aligned} 2^x \equiv\ & 31415926535897932384626433832795028841971693993751058209749 4 \\ & 4592307816406286208998 62 \bmod p. \end{aligned}$$

The right hand side of the congruence consists of the first 84 digits of π.

We found the solution

$$\begin{aligned} x \equiv\ & 756823288306878728158503093002882408211087576743681636958030 \\ & 0654776074817204028 69192 \bmod p-1. \end{aligned}$$

Since the integer -2 is a square modulo p, we may choose the Number Field as $\mathbb{Q}(\sqrt{-2})$. Explicitely, the relations consisted of simultaneously smooth values of

$$\begin{aligned} f_1(X,Y) = & \ 1323274340819980392303558671985532821598359 \cdot X \\ & +8237532479357539733978757237386763941839 67 \cdot Y \end{aligned}$$

and

$$f_2(X,Y) = X^2 + 2Y^2$$

with common root

$$\begin{aligned} (m,1) \equiv\ & (2162322945188147184111028427356103220656020834906357488 \\ & 90531228287292539653562561 1903, 1) \bmod p. \end{aligned}$$

The factor bases were of size 58000 and 11981 respectively.

The sieving procedure was done on 120 workstations using their idle time of about 30.6 mips years. This is faster than using NFS with two quadratic polynomials (44.5 mips years with equal total factor base size). During the sieving step the following amount of partials was collected.

Computing one dependency among the columns of the resulting 119951×69984 linear system modulo q was done by the following steps:

- a compactification step, which resulted in a 51855×51855 system
- the Lanczos algorithm on a Sparc 20 station within 23.2 CPU days using 35 MB of main memory.

Table 6. Collected Partials

#FB	LP	# relations					mips
		full	single	double	triple	quad	years
69981	10^7	15115	136803	335890	280808	59078	30.6

Ideally, the solution of the linear system almost immediately gives the logarithm of all the factor base elements. But we did not get all of them at once, because, for the sake of efficiency, the compactification step removes some relations from the original system. Indeed, applying compactification removed 1088 factor base elements from the linear system. This caused no harm since with the aid of the full relations, the logarithms of these elements could be computed in a negligible amount of time.

In order to be able to compute logs of arbitrary elements, we extended our table of the 69981 factor base logs by creating a data base of 626419 logs of elements with norm up to 10^7.

These were obtained within less than two hours on a Sparc 20 workstation from the partial relations collected during the sieving step.

The log of the element above was derived from the following identities:

$$\begin{aligned}
&314159265358979323846264338327950288419716939937510582097494\\
&4592307816406286208998 62\\
\equiv\ &-11079110202452842718953369487679257497634 03\\
&/12383853456341283587269734548824840418395 9\\
\equiv\ &-7 \cdot 61 \cdot 2594639391675138810059337116552519320289\\
&/13 \cdot 2207 \cdot 3779 \cdot 5053313 \cdot 38665007 \cdot 78959357 \cdot 74034701813 \bmod p.
\end{aligned}$$

We could have set t_1, t_2 to the numerator and the denominator of the first quotient and then proceed with the reduction of section 3. After factoring these two numbers, we observed that all logarithms of the last congruence except from the 40–digit factor

$$p_{40} := 2594639391675138810059337116552519320289$$

and the 11–digit factor

$$p_{11} := 74034701813$$

have already been in our data base. Hence we carried out the reduction step for these two elements.

By applying the reduction step of section 3 (8 hours on Sparc 20), we found that

$$p_{40} \equiv \frac{33613 \cdot 40829 \cdot 83617 \cdot 851761 \cdot 2115961 \cdot 2443219 \cdot 4287211 \cdot 4976687}{2^2 \cdot 19 \cdot 6803 \cdot 8387 \cdot 59387 \cdot 152239 \cdot 586501 \cdot 628997 \cdot 18636193 \cdot 210112139} \bmod p.$$

By sieving (33 minutes on Sparc 20), we found that

$$p_{11} \equiv \frac{-3 \cdot 17 \cdot 37 \cdot 1109 \cdot 6199 \cdot 24989 \cdot 46957 \cdot 120661 \cdot 936667 \cdot 4133219 \cdot p_9}{2^{30} \cdot 5^{29} \cdot 13 \cdot 727 \cdot 1303 \cdot 2399 \cdot 9157 \cdot 32251 \cdot 630299 \cdot 3862493 \cdot 5308663 \cdot p_{9'}} \bmod p,$$

where $p_9 = 515357041$ and $p_{9'} = 422591069$.

The logarithms of a prime number s, with $10^7 < s < 10^{10}$ were found by lattice sieving. In this step, we tested the expressions

$$\frac{f_1(c,d)}{r} \text{ and } f_2(c,d)$$

for smoothness over the factor base, where r ran through the primes of the representations of p_{40} and p_{11} with $r > 719503$. This consumed 1:15 min on a Sparc 20 for each r.

6 Concluding Remarks

Comparing the practical effectiveness of two different index calculus methods for the same prime field is a nontrivial task.

There are aspects, which seem not to be comparable. With the COS method, for instance, you can build a database of logarithms, which considerably simplifies the task of computing individual logarithms. So if one were on the way to break the implementation of a cryptographic protocol, one certainly would prefer the COS method if feasible, even if NFS–DL beat COS. This is because for each individual logarithm, NFS–DL needs solving a large linear system modulo a big prime. In theory, there is even need of repeatedly carrying out the sieving task.

Another aspect is the density of the matrix in the linear algebra step. In our 85–digit example, the comparison has not been a problem, because with COS we found a sparser system in shorter time. It is not clear immediately, how to evaluate the algorithm if we find a sparser system; but we have to pay for that by longer sieving time. The total (real) time here also depends on the number of available workstations that can be used for sieving.

Nevertheless, our comparison of COS and NFS–DL on computing logarithms in $(\mathbb{Z}/p\mathbb{Z})^*$ with a 85–digit p shows that COS is superior for this magnitude of p. A challenging project would be to determine the actual crossover point of COS and NFS–DL.

References

1. D. Coppersmith, A. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica 1*, pages 1–15, 1986.
2. Th. Denny. *Lösen grosser dünnbesetzter Gleichungssysteme über endlichen Primkörpern.* PhD thesis, Universität des Saarlandes/Germany, 1997.

3. Th. Denny and V. Müller. On the reduction of composed relations from the number field sieve. In *Algebraic Number Theory II*, number 1122 in Lecture Notes in Computer Science, 1996.
4. M. Elkenbracht-Huizing. *Factoring integers with the Number Field Sieve.* PhD thesis, Rijksuniversiteit te Leiden, 1996.
5. M. Elkenbracht-Huizing. A multiple polynomial general number field sieve. In *Algebraic Number Theory II*, number 1122 in Lecture Notes in Computer Science, 1996.
6. D. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math.*, 6:124–138, 1993.
7. L. Hua. *Introduction to Number Theory.* Springer, 1982.
8. D. E. Knuth and L. Trabb Pardo. Analysis of a simple factorization algorithm. *Theoretical Computer Science*, 3:321–348, 1976.
9. P. L. Montgomery. Number field sieve with two quadratic polynomials. presentation at Centrum for Wiskunde en Informatica, Amsterdam, 1993.
10. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. on Information Theory*, 24:106–110, 1978.
11. O. Schirokauer. Discrete logarithms and local units. *Phil. Trans. R. Soc. Lond. A 345*, pages 409–423, 1993.
12. O. Schirokauer, D. Weber, and Th. F. Denny. Discrete logarithms: the effectiveness of the index calculus method. In H. Cohen, editor, *Algorithmic Number Theory – ANTS II*, number 1122 in Lecture Notes in Computer Science, 1996.
13. D. Weber. Computing discrete logarithms with the number field sieve. In H. Cohen, editor, *Algorithmic Number Theory – ANTS II*, number 1122 in Lecture Notes in Computer Science, 1996.
14. D. Weber. *On the computation of discrete logarithms in finite prime fields.* PhD thesis, Universität des Saarlandes/Germany, 1997.
15. D. Weber and Th. F. Denny. The solution of McCurley's challenge. 1998. To appear.
16. J. Zayer. *Faktorisieren mit dem Number Field Sieve.* PhD thesis, Universität des Saarlandes/Germany, 1995.