Computing Discrete Logarithms with the General Number Field Sieve

Damian Weber

FB Informatik
Universität des Saarlandes
Postfach 151150
66041 Saarbrücken
Germany
e-mail:dweber@cs.uni-sb.de

Abstract. The difficulty in solving the discrete logarithm problem is of extreme cryptographic importance since it is widely used in signature schemes, message encryption, key exchange, authentication and so on ([15], [17], [21], [29] etc.). The General Number Field Sieve (GNFS) is the asymptotically fastest known method to compute discrete logs mod p [18]. With the first implementation of the GNFS for discrete logs by using Schirokauer's improvement [27] we were able to show its practicability [31].
In this report we write about a new record in computing discrete logarithms mod p and some experimental data collected while finishing the precomputation step for breaking K. McCurley's 129–digit challenge [10].

1 Introduction

Let p be a prime number and $\mathbb{F}_p^*(\cdot)$ be the cyclic multiplicative group of the prime field of p elements, which has order $p-1$. Let $a \in \mathbb{F}_p^*$.
In the case of $b \in \langle a \rangle$, the multiplicative subgroup generated by a, there exist infinitely many $x \in \mathbb{N}_0$ such that

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

holds in $\mathbb{F}_p^*$. We call the minimal $x \in \mathbb{N}_0$ satisfying (1) the *discrete logarithm* of b to the base a.

By $L_p[\nu, \delta]$ we mean the commonly used expression

$$L_p[\nu, \delta] = \exp((\delta + o(1))(\log p)^\nu \cdot (\log\log p)^{1-\nu}).$$

The adaption of the General Number Field Sieve (GNFS) to the discrete log problem was shown by Gordon [18] in 1992 with a conjectured running time

$L_p[\frac{1}{3}, 3^{\frac{2}{3}}]$. This was improved by Schirokauer [27] in 1993 achieving an expected running time of $L_p[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}}]$.

At Eurocrypt 1995 we presented the first practical experiments with the GNFS for discrete logarithms, including a computation of discrete logs of a prime p having 40 decimal digits, where $p-1$ had a 38–digit prime factor [31].

On September 29, 1995, we achieved a new record with a 65–digit p, with $\frac{p-1}{2}$ prime. Furthermore on March 4, 1996, we were able to compute logarithms of factor base elements in the prime field given by the challenge of K. McCurley, which he stated in 1990 in his overview paper [10]. Some practical improvements were necessary to perform the large prime variation as well as to compute individual logarithms.

2 Sketch of the Algorithm

Let a, b as in (1), $q \in \mathbb{N}$ be a prime divisor of $p-1$. We determine the discrete logarithm x of b to the base a modulo q. If we are able to find $s, t \in \mathbb{Z}$, with t coprime to q satisfying the property

$$a^s \cdot b^t = w^q \tag{2}$$

for some $w \in \mathbb{F}_p^*$, then we can compute $x \bmod q$. This is because substituting $b \equiv a^x \bmod p$ leads to

$$x \equiv -st^{-1} \bmod q. \tag{3}$$

So the task is to construct a q–th power in $\mathbb{F}_p^*$, written as a nontrivial product of powers of a and b.
With the GNFS we construct a q–th power in a number ring, the homomorphic image of which is a q–th power in $\mathbb{F}_p^*$.
To generate this number ring, we choose an irreducible polynomial

$$f(X) = a_n X^n + a_{n-1} X^{n-1} + \ldots + a_1 X + a_0. \tag{4}$$

Furthermore, we choose an integer m with $f(m) \equiv 0 \bmod p$ and a rational factorbase $\mathcal{FB}_1$with primes $\in \mathbb{Z}$ at most some $B_1 \in \mathbb{Z}$.
We will compute in the number field $\mathbb{Q}(\alpha)$, where $\alpha \in \mathbb{C}$ a root of f. The ring of integers of $\mathbb{Q}(\alpha)$ is denoted by $\mathcal{O}$. We choose an algebraic factor base $\mathcal{FB}_2$ consisting of all the first degree prime ideals with a norm at most some $B_2 \in \mathbb{Z}$. Because of $f(m) \equiv 0 \bmod p$, the map

$$\begin{aligned} \varphi : \mathbb{Z}[\alpha] &\longrightarrow \mathbb{Z}/p\mathbb{Z} \\ \alpha &\longmapsto m + p\mathbb{Z} \end{aligned}$$

is a ring homomorphism.
We call $l \in \mathbb{Z}$ t–smooth for $t \in \mathbb{Z}$, if all prime divisors of l are at most t.
We identify $\mathbb{F}_p$ with $\mathbb{Z}/p\mathbb{Z}$ and for $a \in \mathbb{F}_p$, we let $\psi(a)$ be the smallest non–negative integer a', such that $a = a' + p\mathbb{Z}$.

The sieving step determines a set S of pairs (c, d) with the following three properties:

- $|S| > |\mathcal{FB}_1| + |\mathcal{FB}_2| + n$
- $c + dm$ is B_1–smooth for every $(c, d) \in S$
- $N(c + d\alpha)$ is B_2–smooth for every $(c, d) \in S$.

In the linear algebra step over $\mathbb{F}_q$, for every $(c, d) \in S$, the Lanczos algorithm determines exponents $e_{c,d}$, such that

- $\prod_S (c + dm)^{e_{c,d}}$ is only divisible by $\psi(a)$ and $\psi(b)$ and
- $\prod_S (c + d\alpha)^{e_{c,d}}$ is a q–th power in $\mathbb{Z}[\alpha]$.

After having accomplished this, we achieve congruence (2) via

$$\begin{aligned} \psi(a)^s \psi(b)^t &\equiv \prod (c + dm)^{e_{c,d}} \\ &\equiv \prod \varphi(c + d\alpha)^{e_{c,d}} \\ &\equiv \varphi(\delta^q) \\ &\equiv d^q \bmod p \end{aligned}$$

for some $\delta \in \mathbb{Z}[\alpha]$ with $\varphi(\delta) = d$. Details about the sieving step are to be found in [19, 32].

The large prime variation, known from the GNFS for factoring, is a standard method to improve the running time in practice [19]. In the current implementation, four large primes are used, therefore we used integers $L_1 > B_1$ and $L_2 > B_2$ as rational and algebraic large prime bounds respectively. Relations containing at least one large prime are called partial relations.

3 Constructing a Polynomial

As described above, there is need to construct a polynomial $f(X) \in Z[X]$ of the form (4) having certain properties. From [27] we know the following conditions:

- f is irreducible
- f is monic
- $f(m) \equiv 0 \bmod p$,
- the constant term of f is a B_2–smooth integer,
- q does not ramify in $\mathcal{O}$ for each prime factor q of $p - 1$ we want to apply the algorithm to.

Let f be given as in (4) and let α be a root of f. We want to construct an algebraic integer, which generates the same field as α over $\mathbb{Q}$. The way to achieve this is already standard when factoring integers with the GNFS [19]. Let

$g(X) = f(\frac{X}{a_n})a_n^{n-1}$, then there exists $\omega \in \mathbb{Q}(\alpha)$ with $g(\omega) = 0$ and $\alpha \cdot a_n = \omega$. As in the case of factoring integers with the GNFS, requiring f to be monic is not really necessary if one uses the identity $\frac{a_n c + d\omega}{a_n} = c + d\alpha$ when the use of an algebraic integer is required. Because of g being monic, ω is an algebraic integer. Furthermore, there is need to get the true decomposition into prime ideals of $\mathcal{O}$ for each $a_n c + d\omega$. We have more freedom in the case of factoring with the GNFS because of the use of arbitrary many quadratic characters there [26], which were discovered by Adleman [1]. The decomposition of the primes of $\mathbb{Z}$ into prime ideals of $\mathcal{O}$ is obtained by finding roots of f mod r, where r is a prime not dividing the index $[\mathcal{O} : \mathbb{Z}[\omega]]$; see for example [8, Th. 4.8.13].

To recognize index divisors r, we search for quadratic divisors of the discriminant of f and apply the Dedekind test to them. The common approach to get polynomials representing p is to compute the m-adic representation of p for several thousand random m's in the interval $[p^{\frac{1}{n+1}}, p^{\frac{1}{n}}]$. Note that for $m \in [\frac{1}{2}p^{\frac{1}{n}}, p^{\frac{1}{n}}]$ you end up with a monic polynomial.
In order to achieve small coefficients, for each m–adic representation of p, we perform an LLL-reduction on the coefficient vectors [20].

Avoiding index divisors restricts the choice of f considerably, but surprisingly this does not prevent us from finding polynomials which lead to elements with small norms. In the following tabular, we list experimental data concerning discrete log problems in $\mathbb{F}_p$, where p has 50, 65 and 75 decimal digits. Here we have examined 4000 polynomials for each p.

digits	poly-type	degree	r^2|disc	good	bad	norm (all)	norm (good)
50	non–monic	3	3844	237	3763	4.18319e+24	4.94237e+24
50	non–monic	4	3944	123	3877	2.20359e+28	7.17503e+28
50	non–monic	5	3927	157	3843	5.11884e+32	1.04916e+33
50	monic	3	2390	2650	1350	8.06197e+29	8.06197e+29
50	monic	4	2555	3024	976	3.2589e+31	3.38198e+31
50	monic	5	2060	2902	1098	6.60776e+34	6.60776e+34
65	non–monic	3	3922	120	3880	9.86409e+28	4.28112e+29
65	non–monic	4	3920	143	3857	1.25294e+31	5.5884e+31
65	non–monic	5	3941	123	3877	9.01888e+34	4.38399e+35
65	monic	3	2915	2391	1609	1.24398e+35	1.2686e+35
65	monic	4	2551	2850	1150	6.70848e+34	6.70848e+34
65	monic	5	2427	2747	1253	1.35132e+38	1.35654e+38
75	non–monic	3	3922	134	3866	2.26212e+31	5.30916e+31
75	non–monic	4	3912	146	3854	1.28099e+33	6.30742e+33
75	non–monic	5	3915	165	3835	5.69116e+36	1.07349e+37
75	monic	3	2540	2432	1568	1.88293e+38	1.88293e+38
75	monic	4	2560	2867	1133	3.12151e+37	3.12151e+37
75	monic	5	2396	2788	1212	1.29024e+40	1.29417e+40

The column poly-type contains the information, whether only monic or non–monic polynomials are considered. The third column lists the degree of the poly-

nomials tested. The next column reports the number of polynomials having square discriminant divisors below the factor base bound. After testing with the Dedekind-criterion, which of the square discriminant divisors are index divisors, we get the number of good polynomials. The others are called bad in the sense, that they cannot be used without having a more time–consuming procedure to recognize the correct exponents in the prime ideal factorization of $a_n c + d\omega$ corresponding to $(c, d) \in S$. As the tabular above shows, the norms are merely slight worse as if the irreducible polynomials could be arbitrary chosen.

4 Constructing the Relation Matrix

The rows of the matrix consist of the exponents of $\mathcal{FB}_1$–elements with respect to $c + dm$, the valuations of the $\mathcal{FB}_2$–prime–ideals with respect to $c + d\alpha$, the approximations of the q–adic logarithms of $a_n c + d\omega$ and the exponent of a_n. As we use large prime relations as described in [32], which we have already generalized to the use of four large primes, we need some utility functions. The relations containing one or more large primes have to be combined to full relations. In the case of factoring integers, this is done via a graph algorithm ([19], [32]). But there are cases which cannot be used in the NFS for discrete log as shown in the following example.

Example. Assume we have three partial relations and the exponents of the three large primes q_1, q_2, q_3 are as follows.

Element	q_1	q_2	q_3
$a_1 + b_1\alpha$	1	1	0
$a_2 + b_2\alpha$	0	1	1
$a_3 + b_3\alpha$	1	0	1

Computing modulo 2 as in the factoring case, we can multiply the three relations and get three large prime squares. But when computing modulo q as in the discrete log case, this does not suffice; in fact the determinant is 2 which is $\not\equiv 0 \bmod q$ for $q > 2$. But in many other cases, the determinant is 0, even in $\mathbb{Z}$.

In the case of the 65–digit–prime, we have 20147 "factoring"–cycles of length ≤ 42 and could combine 19580; that is 97.2%. A similar behaviour we noticed with the McCurley challenge: 37059 cycles with length ≤ 36, and 36840 combined; that is 99.4%. To find the dependencies over $\mathbb{Z}$, we use a very practical method, adopted from the structured Gauss algorithm.

5 Individual Logarithms

For the index calculus algorithms, it is required to reduce the general task of $a^x = b$ to several problems

$$a^x = s,$$

with $\psi(s)$ small. In the Number Field Sieve algorithm the term small means $\psi(s) \leq p^{\frac{1}{n}}$, n the degree of the number field. In the case of finding a relation of the form

$$c + dm \equiv h \cdot \psi(s) \bmod p,$$

where $N(c + d\alpha)$ is B_2–smooth and h is B_1–smooth, we are done.

This is accomplished via a procedure similar to the lattice sieve algorithm [24] and analogous to finding the special relations in [22]. The method we use is as follows. First we find two linear combinations of m and $\psi(s)$

$$dm + y\psi(s) = c \in \mathbb{Z},$$

$$d'm + y'\psi(s) = c' \in \mathbb{Z},$$

where $c, d, c', d', y, y' \leq \sqrt{\psi(s)}$. This is done by using the extended Euclidean algorithm. Set $r := c - dm$, $r' := c' - d'm$, $\beta = c - d\alpha$, $\beta' := c' - d'\alpha$. Since $\psi(s) | \gcd(r, r')$, we can build linear combinations $R(z, z') := zr + z'r' \in \mathbb{Z}$ for small $z, z' \in \mathbb{Z}$. Note that $R(z, z') \equiv 0 \bmod \psi(s)$ and because of the linearity of R, we can use a sieve here. Let $v \in \mathcal{FB}_1$ be a prime of the rational factor base, for fixed z' the prime v divides $R(z, z')$ for

$$z \equiv \frac{z'r'}{z} \bmod v.$$

Exactly the same happens to the prime ideals of the algebraic factor base $\mathcal{FB}_2$. We want to decompose the principal ideals $I(z, z') := (z\beta + z'\beta')$. Let $\mathbf{P}$ be a first degree prime ideal with $N(\mathbf{P}) = v$ corresponding to the root $c_{\mathbf{P}}$ of f modulo v. So $\mathbf{P}$ divides $I(z, z')$ for

$$z \equiv -z' \frac{c' - c_{\mathbf{P}} d'}{c - c_{\mathbf{P}} d} \bmod v.$$

The size of $R(z, z')$ leads to elements $\frac{c+dm}{s}$, with $c, d \leq \sqrt{s}$; so $R(z, z') \leq \frac{1+m}{\sqrt{s}}$. Concerning the size of the norms, we have

$$N(I(z, z')) = \sum_{j=0}^{n} a_j c^j d^{n-j} \leq m \cdot s^{\frac{n}{2}}.$$

6 Computing the Final Result

We describe the computation of the final result modulo q, which is a large prime factor of $p - 1$ by using multiple solutions from the Lanczos algorithm. So the whole computation takes place in the finite field $\mathbb{F}_q$, which we denote by K. Let m be the number of total cycles, i.e. the number of rows of the relation matrix A, r be the number of primes we want to compute the discrete log of. Furthermore, let s be the number of columns with heavy weight, say more than 90 % of the entries are non-zero. Note that $s \geq n$, because the p-adic log

columns are heavy. Then we define the number of the rest of the columns as $t := |FBR| + |FBA| + n + 1 - s - r$.

Then A has the form

$$A = (A'|A''|A''')$$

with $A' \in K^{m \times r}$, $A'' \in K^{m \times t}$, $A''' \in K^{m \times s}$.
We are interested in $r-1$ solutions of

$$x(A''|A''') = 0^T \tag{5}$$

in K.

The Lanczos algorithm computes $s + r - 1$ solutions of $xA'' = 0^T$, which we denote as vectors $s_i := (s_{i1}, \ldots, s_{im})$, for $1 \leq i \leq s + r - 1$. Let $S \in K^{s+r-1 \times m}$ be the matrix consisting of the s_i.

We compute $B = SA''' \in K^{s+r-1 \times s}$ and $r-1$ solutions of

$$xB = 0^T$$

of the form $x_j := (x_{j1}, \ldots, x_{j,s+r-1})$, which we write as a matrix $X \in K^{r-1 \times s+r-1}$.

Then the rows of $XS \in K^{r-1 \times m}$ are solutions to the original equation (5):

$$\begin{aligned} XSA'' &= X0 = 0 \\ XSA''' &= XB = 0 \end{aligned}$$

We now construct linear combinations $XSA' = L \in K^{r-1 \times r}$. For every row i of L, $(1 \leq i \leq r-1)$, we have

$$p_1^{l_{i1}} p_2^{l_{i2}} \cdots p_r^{l_{ir}} \equiv d_i^q \bmod p$$

for some $d_i \in \mathbb{F}_p^*$.

Therefore

$$\begin{aligned} l_{11} \log p_1 + \ldots + l_{1r} \log p_r &\equiv 0 \bmod q \\ \vdots \qquad \vdots \qquad & \vdots \\ l_{r-1,1} \log p_1 + \ldots + l_{r-1,r} \log p_r &\equiv 0 \bmod q \end{aligned}$$

and we need one solution to

$$Ly \equiv 0 \bmod q, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{pmatrix}.$$

Now choose a generator of $\mathbb{F}_p^*$among the p_j, $1 \leq j \leq r$, say p_k and define

$$y' := \frac{1}{y_k} y = \begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_r \end{pmatrix}.$$

We end up with

$$\log_{p_k} p_j \equiv y'_j \bmod q, \qquad 1 \le j \le r \qquad y'_k = 1.$$

7 A 65–digit Problem

We now give details concerning the new record in solving a discrete log problem in a prime field $\mathbb{F}_p$ achieved on September 29, 1995 at the Department of Computer Science in Saarbrücken. The prime number

$$p = 31081938120519680804196101011964261019661412191103091971180537759$$

had 65 decimal digits. We solved

$$7^x \equiv b \bmod p,$$

for $b \in \{2, 3, 5, 11, 13, 17, 19, 23, 29\}$. The prime factorization of $p - 1$ is

$$2 \cdot 15540969060259840402098050505982130509830706095551545985590268879.$$

The polynomial defining the number field was

$$\begin{aligned} f(X) = \quad & -57969887 \quad X^4 \\ & -1040988700418\, X^3 \\ & -1599410033377\, X^2 \\ & +2467898905167\, X \\ & +2804774217242 \end{aligned}$$

The primes of the rational factor base were the primes of $\mathbb{Z}$ less than 27450. The primes of the algebraic factor base were the first degree prime ideals with norm at most 187951, achieving factor base sizes of 3000 and 16954 respectively.
Let α be a root of f. Then the sieving interval for $c + dm, c + d\alpha$ was chosen as follows

$$\begin{aligned} -4000000 \le c \le 4000000 \\ 1 \le d \le 500000. \end{aligned}$$

The sieving procedure used the idle time of 130 workstations and took 5 y 116 d 15 h 36 m (mips) totally. The computation was distributed among them with the Library for Parallel Systems (LiPS), which was developed by our research group [28].

The solution of the 20442×19957 linear system mod q was done on a Paragon machine at the KFA in Jülich/Germany within 38 hours on 50 nodes by using a Lanczos implementation of Th. Denny [12].

The solution of $x \bmod 2$ was easy to obtain by using the Pohlig-Hellman algorithm.

The final step was Chinese remaindering and determined $x \bmod 2q$ with the result

$$
\begin{aligned}
\log_7 2 &= 129474653769238247249574999515030533324375714302685127043203393 44\\
\log_7 3 &= 220801877249312558757608768535153744781711704142180249701754097 92\\
\log_7 5 &= 902036012205447163775276432242132561099089264552949917741405619 6\\
\log_7 11 &= 994555107324467332017738814056228501690291688832819447454410694 2\\
\log_7 13 &= 151567307319432670819633720329050523277239051954853551547690475 94\\
\log_7 17 &= 815265963916162985261666023722445439669180596903218244352067000 7\\
\log_7 19 &= 842943894272204218361130452008301838524298776083122788786982745 8\\
\log_7 23 &= 291530204819307014371483096074026115815057344630346461418287475 93\\
\log_7 29 &= 229979062660061365296829734374134180016821276058004895361687288 07
\end{aligned}
$$

8 Precomputation of the McCurley–Challenge

Using GNFS, we have constructed a linear system for computing factor base logarithms in the prime field presented by K. McCurley [10] in 1990. The problem is as follows. McCurley reports a communication from a Diffie-Hellman-scheme where solving one of two discrete log problems is required. The first one is

$$
\begin{aligned}
7^x \equiv\ & 127402180119973946824269244334322849749382042586931621654557\\
& 735290322914679095998681860978813046595166455458144280588076\\
& 766033781\\
\bmod\ & p
\end{aligned}
\tag{6}
$$

where p is the 129–digit number $p = 2 \cdot 739 \cdot q + 1$, with the prime $q = \frac{7^{149}-1}{6}$.

Let b be the number on the right hand side of (6). Using a combination of trial division and ECM, we were able to reduce the task to computing logs of numbers having at most 26 digits:

$$7^{83} \cdot b \equiv \frac{s}{t} \bmod p$$

$$s := 46197111279788968609389518626845627045463183685525023139231518 95$$
$$t := 44734037033447769048017682021353092450198782047289065217273478 13$$

$$\begin{aligned} s &= 3 \cdot 5 \cdot 11 \cdot 23 \cdot 11287 \cdot 10547587 \cdot 2916781859 \cdot 22761868782949840132373 \\ &\quad \cdot 51337921071904669 \\ t &= 31 \cdot 67 \cdot 7351 \cdot 402869 \cdot 2599909498829 \cdot 3598631011739 \\ &\quad \cdot 77731271923481246820848221 \end{aligned} \tag{7}$$

We chose the number field $Q(\alpha)$, α a root of

$$739X^5 - 5152.$$

The same number field is $Q(\omega)$, where $\omega := 739 \cdot \alpha$ is a root of

$$g(X) := f(\frac{X}{739}) \cdot 739^5 = X^5 - 1536574451494432,$$

ω being an algebraic integer. The set of square divisors of $\mathrm{disc}(g)$ is $\{2, 5, 7, 23, 739\}$. The Dedekind test eventually revealed $2, 5, 739$ as index divisors, but that was not really an obstruction since the prime ideal factorization in $\mathcal{O}$ is the same as in $\mathbb{Z}[\omega]$. We have examined this by using the well known package for computational algebraic number theory PARI [2].

In order to construct a homomorphism $\varphi : \mathbb{Z}[\alpha] \longrightarrow \mathbb{Z}/p\mathbb{Z}$ we set

$$\varphi(\alpha) := 7^{30} = 225393402906922580878632 49.$$

We set the rational large prime bound L_1 to 10^6 and the algebraic large prime bound L_2 to $5 \cdot 10^5$. The sieve range was $-5 \cdot 10^6 \leq c \leq 5 \cdot 10^6$, $1 \leq d \leq 2 \cdot 10^6$

Within 48.5 mips years, we have collected and reduced the following number of relations. The type of relation (smalls, singles, doubles, triples, quadruples) refers to whether the relation contains 0, 1, 2, 3 or 4 large primes respectively. The term filtering means elimination of large primes occurring only once.

type of rel	# after sieving	# after filtering
smalls	2826	2826
singles	37046	32261
doubles	183383	141120
triples	410843	283197
quadruples	332133	210381
total	966231	666959
# of lp	797794	476883

This led to 190077 full relations. As we need only 40000 full relations, we have had a bound of maximal 57 partials per full relation (average 36).

Further sieving, reaching 110.6 mips years, led with the aid of the cycle explosion phenomenon ([16], [32]) to more than 300000 cycles.

type of rel	# after sieving	# after filtering
smalls	3199	3199
singles	42407	38446
doubles	211888	176307
triples	478543	369116
quadruples	388685	282832
total	1124722	869900
cycles	306717	

The large amount of new cycles also has got the advantage of getting much more short cycles – here a new algorithm of combining cycles shows its worth [13]. The resulting matrix is more sparse, we got a maximal cycle length of 22 per full relation (average 15).

Getting the approximations of the q–adic logarithms took 1 min per full relation on a Sparc 20 and was parallelized in a trivial way.

The computation of the 15 solution vectors of the 40015 X 40000 linear system mod q was done on three Sparc 20 stations within a month by using the Lanczos algorithm.

We actually computed the logs of the values of (7), which are in our factor base, for example

$\log_7 67 =$
$1852452768576360356779298469334519947696967677514420577138070596698587572970022624767383788051783719066798580382374055617496228 94$

$\log_7 7351 =$
$1446319689449082956707322689836038098397997958385580595430322834852079492445878021058205245682070769476999126034278024170206345 93$

$\log_7 11287 =$
$7905181248056289435324254076317867160479991300107693131251530114937529830547665373448844397449013712704980829834525824668962996 5.$

The solution of x mod 2 and mod 739 was easily obtained by using the Pohlig-Hellman algorithm with Pollard–Brent improvement ([5], [23], [31]).
Parts of the computation were done by using our package for computational algebraic number theory named LiDIA

`http://www-jb.cs.uni-sb.de/LiDIA/linkhtml/lidia/lidia.html,`

which contains libI as the underlying multiprecision arithmetic [14].

References

1. L. M. Adleman, *Factoring numbers using singular integers*, Proc. 23rd Annual ACM STOC, New Orleans, May 6–8, pp. 64–71, 1991
2. C. Batut, D. Bernardi, H. Cohen, M. Olivier, *GP/PARI CALCULATOR Version 1.39.03*, 1995
3. D. Bernstein, A. K. Lenstra, *A general Number Field Sieve Implementation*, in [19], 1991
4. I. Biehl, J. Buchmann, Th. Papanikolaou *LiDIA – A library for computational number theory*, Universität des Saarlandes, preprint, 1995
5. R. P. Brent, *An Improved Monte Carlo Factorization Algorithm*, Nordisk Tidskrift för Informationsbehandling (BIT) 20, pp. 176–184, 1980
6. J. Buchmann, J. Loho, J. Zayer, *An implementation of the general number field sieve*, Advances in Cryptology Crypto '93 Lecture Notes in Computer Science 773, pp. 159–165, 1993
7. J. P. Buhler, H. W. Lenstra, C. Pomerance, *Factoring integers with the number field sieve*, in [19], 1992
8. H. Cohen, *A course in computational algebraic number theory*, Springer, 1993
9. D. Coppersmith, A. Odlyzko, R. Schroeppel, *Discrete Logarithms in GF(p)*, Algorithmica 1, pp. 1–15, 1986
10. K. McCurley, *The Discrete Logarithm Problem*, Cryptology and Computational Number Theory, Proc. Symp. in Applied Mathematics, American Mathematical Society, 1990
11. Th. Denny, *A Structured Gauss Implementation for GF(p)*, Universität des Saarlandes, to appear
12. Th. Denny, *A Lanczos Implementation for GF(p)*, Universität des Saarlandes, to appear
13. Th. Denny, V. Müller, *On the Reduction of Composed Relations from the Number Field Sieve*, Algorithmic Number Theory Symposium II (ANTS II), 1996
14. R. Dentzer, *libI: eine lange ganzzahlige Arithmetik*, IWR Heidelberg, 1991
15. W. Diffie, M. Hellman, *New directions in Cryptography.* IEEE Trans. Inform. Theory 22, pp. 472-492, 1976
16. B. Dodson, A. K. Lenstra, *NFS with four large primes*, Advances in Cryptology Crypto '95, Lecture Notes in Computer Science 963, Springer, 1995
17. T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory 31, pp. 469-472, 1985
18. D. Gordon, *Discrete Logarithms in GF(p) using the Number Field Sieve*, SIAM J. Discrete Math., Vol 6, pp. 124-138., 1993
19. A. K. Lenstra, H. W. Lenstra, *The development of the number field sieve*, Springer, 1993
20. A. K. Lenstra, H. W. Lenstra, M. S. Manasse, J. M. Pollard, *The number field sieve*, Abstract: Proc. 22nd Ann. ACM Symp. on Theory of Computing (STOC), 564-572, 1990
21. National Institute of Standards and Technology. *The Digital Signature Standard, proposal and discussion*, Comm. of the ACM, 35 (7), pp. 36-54, 1992
22. A. Odlyzko, M. LaMacchia, *Discrete Logarithms in GF(p)*, 1991
23. J. M. Pollard, *Monte Carlo Methods for Index Computation (mod p)*, Math. Comp. 32, 918–924, 1978
24. J. M. Pollard, *The lattice sieve*, in [19], 1991

25. S. Pohlig, M. Hellman, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Trans. on Inform. Theory 24, 106–110, 1978
26. O. Schirokauer, personal communication, 1995
27. O. Schirokauer, *Discrete Logarithms and Local Units*, Phil. Trans. R. Soc. Lond. A 345, 409–423, 1993
28. Th. Setz, R. Roth, *LiPS: a System for Distributed Processing on Workstations*, SFB 124 TP D5, Universität des Saarlandes, 1992
29. D. R. Stinson, *Cryptography in Theory and Practice*, CRC Press, 1995
30. D. Shanks, *Class Number, a Theory of Factorization and Genera*, Proc. Symposium Pure Mathematics Vol. 20, American Mathematical Society, Providence, R. I., pp. 415–440, 1970
31. D. Weber, *An Implementation of the Number Field Sieve to Compute Discrete Logarithms mod p*, Advances in Cryptology – Eurocrypt'95, Lecture Notes in Computer Science 921,pp. 95–105, 1995
32. J. Zayer, *Faktorisieren mit dem Number Field Sieve*, PhD thesis, Saarbrücken, 1995