An Implementation of the General Number Field Sieve to Compute Discrete Logarithms mod p

Damian Weber

FB Informatik
Universität des Saarlandes
Postfach 151150
66041 Saarbrücken
Germany

Abstract. There are many cryptographic protocols the security of which depends on the difficulty of solving the discrete logarithm problem ([8], [9], [14], etc.). In [10] and [18] it was described how to apply the number field sieve algorithm to the discrete logarithm problem in prime fields. This resulted in the asymptotically fastest known discrete log algorithm for finite fields of p elements. Very little is known about the behaviour of this algorithm in practice. In this report we write about our practical experience with our implementation of their algorithm whose first version was completed in October 1994 at the Department of Computer Science at the Universität des Saarlandes.

1 Introduction

The importance of the Discrete Logarithm Problem has its roots in its cryptographic significance. Many protocols in cryptography, for example the Digital Signature Standard [14], are secure if the underlying Discrete Logarithm Problem is difficult to solve.
A lot of algorithms have already been created to find a solution to it and therefore to break one cryptosystem associated with it.

There is an early method which proves to be quite successful for groups of smooth orders, i.e. which have no large prime factor. It was published by Pohlig and Hellman [17] and independently by Silver. It can be improved by an idea of Shanks [19]. We actually use the very practical improvement of Pollard [16].
The first of the class of index calculus algorithms to which the algorithm we discuss belongs to was published by Kraitchik and Cunningham and later rediscovered and analyzed by Adleman, Merkle and Pomerance [5]. It has a running time of $L_p[\frac{1}{2}, \delta]$ for some $\delta > 0$. With $L_p[\nu, \delta]$ we mean the commonly used expression

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

There are variations of this index calculus algorithm, discovered by Coppersmith, Odlyzko and Schroeppel [4] with conjectured running time $L_p[\frac{1}{2}, \delta]$.

The first algorithm with expected running time $L_p[\frac{1}{3}, 3^{\frac{2}{3}}]$ was detected by Dan Gordon [10] in 1992. This was improved by Oliver Schirokauer [18] in 1993 achieving an expected running time of $L_p[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}}]$.

It is based on the Number Field Sieve, a method which has already been used to factor integers ([1],[2]).

If this algorithm proves to be valuable in practice, the security parameters of much implemented cryptosystems have to be thought over. In this report we write about our practical experience with our first implementation of their algorithm whose first version was completed in October 1994 at the Department of Computer Science at the Universität des Saarlandes.

For our implementation we used the methods described in [10] and [18]. After a short description of their algorithm we show how the general problem can be treated conveniently. Furthermore we consider the running time in practice, and give some impressive results concerning the comparison with our implementation of the algorithm of Pohlig and Hellman, which has an expected running time of $L_p(1, \frac{1}{2})$. Clearly, as the algorithm of Pohlig and Hellman is not an index calculus method, there is need for a comparison with an implementation of the methods of Coppersmith, Odlyzko and Schroeppel [4].

2 The Discrete Logarithm Problem in $\mathbb{F}_p$

We consider $\mathbb{F}_p(\cdot)$, the cyclic multiplicative group of the prime fields of p elements, p prime, which has order $p-1$.
Let $a, b \in \mathbb{F}_p$.
If there exists $x \in \mathbb{N}_0$ such that

$$a^x = b,$$

we define the least such $x \in \mathbb{N}_0$ as the *discrete logarithm* of b to the base a.

3 The General Number Field Sieve (GNFS)

With $a, b \in \mathbb{F}_p$, we determine the discrete logarithm x of b to the base a modulo $q \in \mathbb{N}$ where q is a prime divisor of $p-1$. Then we combine the results for every q dividing $p-1$ via the Chinese remainder algorithm. In order to determine the discrete logarithm x modulo q, we use the GNFS to construct q–th powers in $\mathbb{F}_p$.
If we are able to find integers s, t with the property

$$a^s \cdot b^t \equiv w^q \bmod p$$

for some $w \in \mathbb{F}_p$, and a rational integer q coprime to t, then we have computed $x \bmod q$. If this is the case, writing $b \equiv a^x \bmod p$ leads to

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

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.
First, choose an irreducible polynomial

$$f(X) = X^n + a_{n-1}X^{n-1} + \ldots + a_1X + a_0,$$

an integer m and a rational factorbase $\mathcal{FB}_1$.
We denote by K the field $\mathbb{Q}[\alpha]$. We will work in the ring of integers $\mathcal{O}_K \subset K$, α a zero of f in $\mathbb{C}$.
We choose an algebraic factor base $\mathcal{FB}_2$ consisting of first degree prime ideals with norm less than some bound.
For a set M of integers and an integer l we say that l is *M–smooth*, if all the prime divisors of l lie in M. We call l to be m–smooth for an integer m, l is $\{1, \ldots, m\}$–smooth.

After the choice of the polynomial f and the factor bases $\mathcal{FB}_1$ and $\mathcal{FB}_2$, the following conditions on f must be satisfied:

- $f(m) \equiv 0 \bmod p$,
- $m = h \cdot b$ where h is a $\mathcal{FB}_1$–smooth integer,
- p does not ramify in $\mathcal{O}_K$,
- the constant term of f is a $\mathcal{FB}_2$–smooth integer,

- p does not divide the discriminant of f,
- q does not ramify in $\mathcal{O}_K$ for each divisor q of $p-1$ we want to apply the algorithm to.

Because of the first condition the map

$$\begin{aligned} \varphi : \mathbb{Z}[\alpha] &\longrightarrow \mathbb{F}_p \\ \alpha &\longmapsto m \end{aligned}$$

is a ring homomorphism.

The algorithm determines a non empty set S of pairs (c, d) with the following property:

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

Therefore, the following congruence holds.

$$\begin{aligned} \prod (c + dm)^{e_{c,d}} &= \prod \varphi(c + d\alpha)^{e_{c,d}} \\ &= \varphi(\omega^q) \\ &\equiv w^q \bmod p \end{aligned}$$

It is clear immediately that $\prod_S (c + dm)^{e_{c,d}}$ is a q–th power in $\mathbb{F}_p$.

4 The Sieving Stage

In the sieving stage we collect pairs (c, d) of integers for which

- $c + dm$ is $\mathcal{FB}_1$–smooth
- $c + d\alpha$ is $\mathcal{FB}_2$–smooth.

Each pair which satisfies these conditions we call a *hit*.

If we have more than $|\mathcal{FB}_1| + |\mathcal{FB}_2|$ hits collected, a solution of a linear system $\bmod q$ leads to an equation

$$\begin{aligned} \prod (c + dm) &= \prod \varphi(c + d\alpha) \\ &= \varphi(\omega^q) \\ &\equiv w^q \bmod p \end{aligned}$$

where the left side is only divisible by a and b.

In the case of a, b being primes, we have $a^s b^t \equiv w^q \bmod p$ as desired. If a, b are not primes, we get a small linear system modulo q. It is convenient to avoid this by using the reduction we describe in section 6.

5 Constructing q-th powers in $\mathcal{O}_K$

In the previous section the construction of a q-th power in $\mathcal{O}_K$ is required. The details of this construction are to be found in [18]. In the following we give a brief overview.

Let

$$q\mathcal{O}_K = \prod_{\rho=1}^{r} \pi_\rho$$

be the decomposition of q into prime ideals of $\mathcal{O}_K$ and

$$\epsilon = \operatorname{lcm}_\rho \{N(\pi_\rho) - 1\}.$$

It follows that

$$\gamma^\epsilon \equiv 1 \bmod \mathcal{O}_K/\pi_\rho \qquad \text{for } \gamma \in \mathcal{O}_K/\pi_\rho \text{and } 1 \le \rho \le r.$$

Define λ to be the following map.

$$\begin{aligned} \lambda : (\mathcal{O}_K, \cdot) &\longrightarrow q\mathcal{O}_K/q^2\mathcal{O}_K(+) \\ \gamma &\longrightarrow \gamma^\epsilon - 1 \end{aligned}$$

Because of q not being ramified in $\mathcal{O}_K$, this is actually a homomorphism of semi groups and a homomorphism on the group of units of $\mathcal{O}_K$.
We consider a special case of the main result of [18].

Proposition 1. *Let γ be an element of $\mathcal{O}_K$ whose norm is not divisible by q. Let U be the group of units of $\mathcal{O}_K$. Let*

$$U' = \{\eta \in U \mid \eta \equiv 1 \ mod\ q\mathcal{O}_K\}.$$

Then γ is a q–th power in $\mathcal{O}_K$, if

i) the class number of K is not divisible by q,
ii) $U' \subset U^q$,
iii) $ord_Q(\gamma) \equiv 0 \ mod\ q$ for all prime ideals Q of $\mathcal{O}_K$,
iv) $\lambda(\gamma) = 0$.

For each pair (c, d) which we have detected as a hit we compute the image under λ modulo q^2 using the α–power basis of $\mathbb{Z}[\alpha]/q^2\mathbb{Z}[\alpha]$.

$$\lambda(c + d\alpha) = \sum_{j=0}^{n-1} b_j\alpha^j \bmod q^2\mathcal{O}_K.$$

We aim

$$\sum \lambda(c + d\alpha) = 0 \bmod q^2.$$

But all the $\lambda(c + d\alpha)$ are multiples of q. Therefore we can divide each b_j by q and then take the sum $\bmod q$ instead of computing $\bmod q^2$.
With this argument we supply to the exponent vector of the prime ideals the coefficients b_j of the image under λ. This means the exponent vector gets extended by n entries.
This concludes the construction.

6 A Reduction of the General Problem

It is convenient to transform the original discrete logarithm problem into an easier one, which means the numbers a, b are prime and smaller than a given bound. We require the following conditions on a and b and we will show how to change the original task appropriately.

Condition 1: a shall be a prime $\in \mathcal{FB}_1$
Condition 2: b shall be a prime $\leq \sqrt[n]{p}$

We give a brief description how this can be achieved.

1. We factor $p - 1 = \prod_{i=1}^{r} q_i^{e_i}$ using the elliptic curve method because this method is fast enough for numbers having less than 40 digits.
2. We find a generator g modulo p, which is in our rational factor base $\mathcal{FB}_1$.
3. We find l such that $a^l \cdot b \equiv c \mod p$ is $\sqrt[n]{p}$–smooth
$$c = \prod s_i^{e'_i}.$$
4. For every i we solve $g^{x_i} \equiv s_i \mod p$:
   (a) using the GNFS ($m = h \cdot s_i$, h $\mathcal{FB}_1$–smooth) we find a relation
   $$g^{y_j} \cdot s_i^{y'_j} \equiv d_i^{q_j^{e_j}} \mod p$$
   for $1 \leq j \leq r$,
   (b) it follows that $x_i \equiv -\frac{y_j}{y'_j} \mod q_j^{e_j}$.
5. We solve $g^z \equiv a \mod p$, $z \equiv z_j \mod q_j^{e_j}$ and compute
$$\log_a b \equiv \frac{x_i}{z_i} - l \mod q_i^{e_i}$$

7 A 25–digit example

The first example which could not be done with our implementation of the Pohlig–Hellman procedure was the following.

We solved
$$7^x = 17 \bmod p,$$
where p is the 25-digit number 1234567890123456789000421.

Factoring $p - 1$ by trial division equals
$$1234567890123456789000421 - 1 = 2^2 * 3 * 5 * q,$$
where q is the 23–digit prime number 20576131502057613150007.

Since $7^{\frac{p-1}{q'}} \not\equiv 1 \bmod p$ for $q' \in \{2, 3, 5, q\}$, 7 is a generator of $\mathbb{F}_p$.
Because of the 23–digit prime factor q in the factorization of $p-1$, our implementation of the methods of Pohlig–Hellman–Pollard was not successful within 96 hours.
So we started our GNFS algorithm by using the polynomial

$$f(X) = X^3 + 57007X^2 - 27942X - 31727$$

and set $m = 107257599$ with $f(m) \equiv 0 \bmod p$.
As $b = 17$ is an element of our rational factor base there is no need of satisfying condition 3 of section 6.
The primes of both factor bases were the first-degree primes with norm less than 2400; no large primes were used.
The sieving interval for $c + dm, c + d\alpha$ was chosen as follows:

$$\begin{aligned} -4000000 \leq c &\leq 4000000 \\ 1 \leq d &\leq 500. \end{aligned}$$

The sieving procedure was performed on a Sparc ELC workstation with 16 MB RAM within 12 hours.
The solution of the 728×703 linear system $\bmod q$ was done on a Sparc ELC workstation with 64 MB RAM within 8 hours.
The solutions of $x \bmod 2^2, 3, 5$ were easily to obtain by using the Pohlig–Hellman–Shanks procedure.
In the final step we had to combine the results by using the Chinese–Remainder–Algorithm and

$$x = 256351350915151146893061 \bmod 1234567890123456789000420$$

was established.

8 A 40–digit example

At February 2, 1995, the solution of our second interesting example has been achieved with the aid of Zayer's implementation of the General Number Field Sieve, which has already been used successfully to factorize a 70–digit number and to sieve in the case of a 107–digit number [2].

Here we solved

$$23^x = 29 \bmod p,$$

where p is the 40-digit number 3108193812051968080419611909199224122909.

Factoring $p - 1$ by trial division equals

$$3108193812051968080419611909199224122909 - 1 = 2^2 * 3^2 * q,$$

where q is the 38–digit prime number 86338717001443557789433664144422892303.

Again $23^{\frac{p-1}{q'}} \not\equiv 1 \bmod p$ for $q' \in \{2, 3, q\}$, and 23 is a generator of $\mathbb{F}_p$.
The GNFS algorithm has been started by using the polynomial

$$f(X) = X^3 + 67025431X^2 + 3599000298704X - 6293411590817$$

and the integer $m = 14593810375959$ with $f(m) \equiv 0 \bmod p$.
Again $b = 29$ is an element of our rational factor base, so there is no need to satisfy condition 3 of section 6.
The primes of the rational factor base are the 1493 primes $p \leq 12503$.
The primes of the algebraic factor base are the 1978 first-degree prime ideals with norm less than 17321. The sieving interval for $c + dm, c + d\alpha$ was chosen as follows

$$-5000000 \leq c \leq 5000000$$
$$1 \leq d \leq 5000.$$

The sieving procedure was done on a Sparc ELC workstation with 16 MB RAM within 21 hours.
The solution of the 3500×3477 linear system $\bmod q$ was obtained on a Paragon massively parallel system at KFA in J"ulich within 40 minutes on 60 nodes. The task was carried out by a structured Gauss implementation of Thomas Denny [6], which uses the LIP package of Arjen Lenstra [13].
The solutions of $x \bmod 2^2, 3, 5$ were easily obtained by using the Pohlig–Hellman–Shanks procedure.
In the final step we had to combine the results using the Chinese-Remainder-Algorithm and

$$x = 17611497414534741323045752017156439409 20$$
$$\bmod 310819381205196808041961190919922412 2908$$

was established.

9 Experimental Results of the Pohlig–Hellman Algorithm

The algorithm of Pohlig and Hellman works well in the case of q being small. We want to know how small the biggest prime factor q should be so that compared with the GNFS implementation this algorithm is faster. Our experimental results show that there is no reason to work with it if $q \geq 10^{12}$.
As above, the computations have been done on a Sparc ELC workstation with 16 MB RAM (21 Mips).
As the running time of both algorithms depends on the largest prime factor of $p - 1$, we have used the hardest primes p, namely those primes where $\frac{p-1}{2}$ is a prime, too. So the amount of time needed to solve the whole problem can be viewed as the time to determine the solution in the subgroup of quadratic residues $\bmod p$ which has order $\frac{p-1}{2}$.

We consider twenty discrete log problems, ten with primes of thirteen decimal digits and ten with primes of fifteen decimal digits.

DL–Problem	Running time
$2^x \equiv 5 \mod 2000000000123$	6 min 25 sec
$2^x \equiv 5 \mod 2000000001443$	45 min 58 sec
$13^x \equiv 17 \mod 2000000001599$	36 min 40 sec
$5^x \equiv 7 \mod 2000000002487$	19 min 10 sec
$11^x \equiv 13 \mod 2000000003879$	99 min 7 sec
$2^x \equiv 5 \mod 2000000004347$	18 min 2 sec
$2^x \equiv 5 \mod 2000000007107$	22 min 2 sec
$2^x \equiv 5 \mod 2000000007683$	41 min 14 sec
$5^x \equiv 10 \mod 2000000007767$	15 min 33 sec
$5^x \equiv 7 \mod 2000000008367$	24 min 56 sec
$2^x \equiv 5 \mod 100000000005083$	84 min 22 sec
$5^x \equiv 10 \mod 100000000005527$	307 min 34 sec
$5^x \equiv 7 \mod 100000000007807$	354 min 38 sec
$5^x \equiv 10 \mod 100000000008863$	215 min 15 sec
$13^x \equiv 19 \mod 100000000010279$	186 min 33 sec
$2^x \equiv 5 \mod 100000000012307$	117 min 59 sec
$2^x \equiv 5 \mod 100000000013027$	262 min 48 sec
$17^x \equiv 23 \mod 100000000015439$	557 min 8 sec
$2^x \equiv 5 \mod 100000000016747$	197 min 43 sec
$2^x \equiv 6 \mod 100000000017899$	250 min 30 sec

This is an average running time of 32 min. 55 sec. for the 13–digit primes and an average running time of 253 min. 27 sec for the 15–digit primes.

10 GNFS versus Pohlig–Hellman–Algorithm

We have solved the problems of the section before with our Number field sieve implementation.
We start with our choice for the 13–digit primes p. Here we have chosen a polynomial of degree 2 and $m = 1388297$.
If one takes $m = \sqrt{p}$ instead, the average norm of $c + \alpha$ is $3.38 \cdot 10^9$, which is slightly larger compared to our choice of m, where we got $1.21 \cdot 10^9$.
Each factor base is bounded by a value between 200 and 300, so we expect to get between 100 and 140 factor base elements totally.
So the polynomials are determined by m

$$f(X) = X^2 + 52317X + (p - m^2 - 52317m).$$

We have allowed one large prime for each of the two factor bases. The large prime bound for $\mathbb{Z}$ is 2000000 and for $\mathcal{O}_K$ it is 4000000. A description of the large prime variation can be found in [11].
The choice of the parameters in the case of the 15–digit primes was quite similar.

With $m = 9964218$ we have an average norm of $9.9 \cdot 10^8$ instead of $3.4 \cdot 10^9$ with $m = \sqrt{p}$.
Here the polynomials are

$$f(X) = X^2 + 71692X + (p - m^2 - 52317m).$$

The large prime bounds are chosen as above.
In the following table we list the number of full relations, i.e. relations without large primes as well as the number of single large prime relations with either a large rational prime or a prime ideal with large norm and the number double large prime relations, i.e. a rational single large prime and an algebraic single large prime.

We have listed the running times of the sieve and the solution of the linear system mod q.

Prime	full	single	double	Sieve	LS	total
2000000000123	63	95	48	02:51	01:48	04:39
2000000001443	133	0	0	04:29	02:14	06:43
2000000001599	98	105	63	03:07	01:46	04:53
2000000002487	51	70	56	03:08	01:53	05:01
2000000003879	35	42	26	03:47	01:19	05:06
2000000004347	33	70	37	05:05	02:20	07:25
2000000007107	40	67	39	05:42	02:46	08:27
2000000007683	174	303	122	05:24	03:46	09:10
2000000007767	61	96	41	07:03	02:22	09:25
2000000008367	55	85	28	05:24	03:36	09:00
100000000005083	72	123	53	05:32	04:18	09:50
100000000005527	111	148	50	03:34	03:38	07:12
100000000007807	80	87	49	03:24	08:17	11:41
100000000008863	124	87	54	03:57	08:41	12:37
100000000010279	114	143	51	03:14	03:43	06:57
100000000012307	68	145	48	06:38	06:04	12:42
100000000013027	95	86	54	05:02	06:04	11:06
100000000015439	104	110	64	04:38	04:20	08:58
100000000016747	137	103	0	04:01	03:47	07:48
100000000017899	50	102	35	07:42	02:45	10:27

This is an average running time of 6 min. 59 sec. for the 13–digit primes and an average running time of 9 min. 56 sec for the 15–digit primes.
All the integer computations were performed by using the libI package of Ralf Dentzer [7], which we have embedded in a C++ class library called LiDIA [15].

References

1. D. Bernstein, A. K. Lenstra, *A general Number Field Sieve Implementation* , in [11], 1991
2. J. Buchmann, J. Loho, J. Zayer, *An implementation of the general number field sieve* , Advances in Cryptology Crypto (1993) Lecture Notes in Computer Science 773, pp. 159–165
3. J. P. Buhler, H. W. Lenstra, C. Pomerance, *Factoring integers with the number field sieve*, in [11], 1992
4. D. Coppersmith, A. Odlyzko, R. Schroeppel, *Discrete Logarithms in GF(p)* , Algorithmica 1, 1986, pp. 1–15
5. K. Mc Curley, *The Discrete Logarithm Problem*, Cryptology and Computational Number Theory, Proc. Symp. in Applied Mathematics, American Mathematical Society, 1990
6. Th. Denny, *A Structured Gauss Implementation for GF(p)*, Universität des Saarlandes, to appear
7. R. Dentzer, *libI: eine lange ganzzahlige Arithmetik*, IWR Heidelberg, 1991
8. W. Diffie, M. Hellman, *New directions in Cryptography.* IEEE Trans. Inform. Theory 22 (1976), pp. 472-492
9. T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory 31 (1985), pp. 469-472
10. D. Gordon, *Discrete Logarithms in GF(p) using the Number Field Sieve*, University of Georgia, preprint 1992
11. A. K. Lenstra, H. W. Lenstra, *The development of the number field sieve*, Springer–Verlag, 1993
12. 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)(1990),564-572
13. A. K. Lenstra, *lip: A long integer package*, Bellcore, 1989
14. National Institute of Standards and Technology. *The Digital Signature Standard, proposal and discussion*, Comm. of the ACM, 35 (7), pp. 36-54, 1992
15. I. Biehl, J. Buchmann, Th. Papanikolaou *LiDIA – A library for computational number theory*, Universität des Saarlandes, submitted to ISAAC 1995
16. J. M. Pollard, *Monte Carlo Methods for Index Computation (mod p)* , Math. Comp. 32, 918–924, 1978
17. 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
18. O. Schirokauer, *Discrete Logarithms and Local Units*, Phil. Trans. R. Soc. Lond. A (1993) 345, 409–423
19. D. Shanks, *Class Number, a Theory of Factorization and Genera* , Proc. Symposium Pure Mathematics Vol. 20, American Mathematical Society, Providence, R. I., 1970, pp. 415–440

This article was processed using the LaTeX macro package with LLNCS style