Discrete Logarithms: Recent Progress

Johannes Buchmann[1], Damian Weber[2]

[1] Technische Universität Darmstadt
Alexanderstraße 10
D–64287 Darmstadt
buchmann@cdc.informatik.tu-darmstadt.de

[2] Institut für Techno– und Wirtschaftsmathematik
Erwin–Schrödinger–Str. 49
D–67663 Kaiserslautern
weber@itwm.uni-kl.de

Abstract

We summarize recent developments on the computation of discrete logarithms in general groups as well as in some specialized settings. More specifically, we consider the following abelian groups: the multiplicative group of finite fields, the group of points of an elliptic curve over a finite field, and the class group of quadratic number fields.

Keywords: Discrete Logarithms, Groups, Cryptography

1 Introduction

Within the last few years, due to applications in cryptography, an enormous interest has grown in the question of the actual difficulty of the discrete logarithm (DL) problem in groups. In this paper we summarize recent developments concerning this computational problem.

On the one hand, cryptographers want to find finite groups where this problem is presumably hard – or even better – provably hard. On the other hand, competition arises due to the research in the area of computational number theory where efforts are made to develop efficient algorithms to solve such problems, and exhibiting weaknesses of the corresponding cryptographic protocols.

Since the proposal of the Diffie–Hellman key exchange protocol [13], several other protocols have been developed, whose security depends on the difficulty in solving the DL problem. The DL problem for a group G may be stated as follows.

Given a, $b \in G$, find an $x \in \mathbb{Z}$, such that

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

or prove that such an x does not exist. If x exists, we call the minimal non-negative solution of (1) the discrete logarithm of b with respect to a. We will stick to this notation throughout this article.

Motivated by the existence of subexponential algorithms for $G = GF(p)^*$, the designers of new cryptosystems incorporate other groups into their protocols, in order to avoid subexponential attacks. Though not possible in any case, sometimes the designer may use an arbitrary group with the only precondition that the DL problem be hard (for example the zero–knowledge protocol for DL from [8, 9]). Among these are the group of points of an elliptic curve over a finite field [15] as well as the Jacobian group of a hyperelliptic curve [16]. For none of these a subexponential algorithm is known. For some special cases, which are easily avoided in cryptographic applications, however, faster algorithms are known. This is the case for elliptic curves of trace 1 [30], hyperelliptic curves of large genus [1] and class groups of imaginary quadratic number fields [4].

When trying to solve the DL problem, a first approach is to use generic algorithms which work in any group and use only group operations (multiplication, inversion, equality testing). In section 2 we review both a deterministic and a probabilistic algorithm which produce a solution of (1) after at most $O(\sqrt{|G|})$ group operations. As we will see, both algorithms are optimal in the sense that $O(\sqrt{|G|})$ is shown to be a lower bound for generic algorithms.

In section 3 the group of points of elliptic curves over $GF(p)$ and $GF(2^n)$ is considered. We summarize the results of the probably largest effort to date to attack a DL problem via the implementation of a generic algorithm.

Algorithms of subexponential type can be found if there is an efficient way to produce relations among group elements. That means to form non–trivial power products of elements of a small set which evaluate to the unity element of the group. This is the case, for example, in finite prime fields. The recent practical progress in this setting is surveyed in section 4.

2 Generic Algorithms

Let G be a finite abelian group. In this section we consider algorithms which, given two elements $g, g' \in G$, only make use of three types of operations:

- computing $gg' \in G$,
- computing $g^{-1} \in G$,
- deciding whether $g = g'$.

These are referred to as *group operations*. We denote the identity of G by 1.

2.1 Reducing to Cyclic Groups

It is well known that there are positive integers $m_1, \dots, m_k$, $k \geq 1$ – the *invariants* of G – where m_i divides m_{i+1} for $1 \leq i < k$ such that

$$G \cong \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}. \tag{2}$$

The DL problem in the group on the right hand side of (2) can be reduced to the DL problem in each of the $\mathbb{Z}/m_i\mathbb{Z}$. With $m := m_i (1 \leq i \leq k)$, solving the DL problem in $\mathbb{Z}/m\mathbb{Z}$ means solving the congruence

$$ax \equiv b \bmod m$$

which can be done in polynomial time by means of the Euclidean algorithm. At first sight, this suggests that the DL problem is easy in general since (2) describes the structure of any finite abelian group. The problem, however, is that in general neither the invariants of G nor the isomorphism of (2) is known. The difficulty of computing discrete logarithms in G is therefore closely related to the difficulty of finding the invariants of G and the isomorphism of (2).

If computing the invariants is possible, the problem can be reduced further to groups of prime order as we shall see in the next section.

2.2 Reducing to Prime Order Groups

We recall a method by Silver, Pohlig and Hellman, which first has been described for $GF(p)^*$, when $p-1$ is smooth. This method works for arbitrary cyclic groups G of order n and can be used to reduce the DL problem in G to DL problems in prime order groups G_r, where $r|n$.

Let r be any prime dividing n and h an integer with $h \geq 1$. Let r^h be a power of r dividing n; then we are going to compute x modulo r^h.

Suppose (1) is solvable. If $h = 1$, solve $a^{x_0} = b \mod G/G^r$. This group has prime order r. Then $a^{x_0} = a^x c^r$ for some $c \in G$. If g is a generator of G with $g^l = a, g^k = c$, we obtain $lx_0 = lx + kr$ which is equivalent to $x \equiv x_0 \mod r$.

Assume now that we know the value of $x \mod r^{h-1}$, written as

$$x \equiv x_0 + x_1 r + x_2 r^2 + \cdots + x_{h-2} r^{h-2} \bmod r^{h-1}.$$

Set

$$a' = a^{(p-1)/r},$$
$$b' = \left(\frac{b}{a^{x_0+x_1r+x_2r^2+\cdots+x_{h-2}r^{h-2}}}\right)^{(p-1)/r^h}.$$

Because of

$$a'^r = 1 \quad \text{and}$$
$$b'^r = 1 \bmod p$$

both a' and b' are members of the unique subgroup G_r of order r in G.

Solving

$$a'^{x_{h-1}} = b' \quad \text{for } x_{h-1},$$

we obtain

$$a'^{x_{h-1}} = \left(\frac{b}{a^{x_0+x_1r+x_2r^2+\cdots+x_{h-2}r^{h-2}}}\right)^{(p-1)/r^h}.$$

With

$$a'^{x_{h-1}} = \left(a^{x_{h-1}}\right)^{\frac{p-1}{r}} = a^{r^{h-1}x_{h-1}\frac{p-1}{r^h}},$$

it follows that

$$\left(a^{x_0+x_1r+x_2r^2+\cdots+x_{h-1}r^{h-1}}\right)^{\frac{p-1}{r^h}} = b^{\frac{p-1}{r^h}},$$

which is equivalent to

$$x \equiv x_0 + x_1 r + x_2 r^2 + \cdots + x_{h-2} r^{h-2} + x_{h-1} r^{h-1} \bmod r^h.$$

2.3 Shanks's Baby–Step–Giant–Step Algorithm

The first algorithm, which is deterministic and runs in time $O(\sqrt{n})$ is Shanks's Baby–Step–Giant–Step algorithm [27]. Let $m = \lfloor \sqrt{n} \rfloor + 1$, such that $x = x_1 m + x_2$ with (unknown) $x_1, x_2 < m$. After computing a set $M := \{a^{mi} \mid 0 \leq i \leq m\}$ (the *giant steps*), it remains to decide whether one of the elements $a^{-j}b$, $0 \leq j < m$ lies in M (the *baby steps*). If a match is found, $a^{mi} = a^{-j}b$, therefore $x := mi + j$ satisfies (1). Otherwise, b is not contained in the subgroup generated by a. Obviously, this algorithm requires at most $T(n) = 2\sqrt{n} + T'(\sqrt{n})\sqrt{n}$ group operations, where $T'(m)$ is the time to decide membership for a given element in M. Linear search in M would cause $T(n) > n$, which is slower than the trivial algorithm. With a total ordering of the representation of group elements, however, one may sort M and do binary search on M. There are sorting algorithms which sort m elements within time $O(m \log m)$ (for example heapsort [11]); binary search on m elements consumes time $O(\log m)$. Summing up, we obtain

$$T(n) = C\sqrt{n} \log n$$

for some constant C, which is $O(n^{1/2+\epsilon})$ for all $\epsilon > 0$. Because M has to be stored, this algorithm also consumes $O(\sqrt{n})$ space. We note that there is a time/memory trade–off by reducing the table size $m < \sqrt{n}$ and carrying out $n/m > \sqrt{n}$ membership tests.

By refining this method a considerable theoretical and practical improvement is achieved by Buchmann, Jacobson and Teske (BJT) in [5]. The authors extract the discrete logarithm in time $O(\sqrt{x} + \log \sqrt{x})$ by using a table of $O(\sqrt{x})$ entries. Note that the running time depends on the (unknown) discrete logarithm itself. In the case that the DL does not exist, the running times and space requirements hold for $x := \mathrm{ord}_G a$. In their paper also the practical significance is illustrated on the special case $G = Cl(\Delta)$ of the ideal class group of imaginary quadratic orders. The largest example is given for $\Delta = -4 \cdot (10^{20} + 1)$ where G is the subgroup generated by the ideal over 7 which consists of 1 856 197 104 elements. Clearly, the smaller x the performance compared to Shanks's original algorithm is the better, as can be seen from table 1.

2.4 Pollard's Probabilistic Algorithms

The advantage of Pollard's algorithms is the use of constant space while preserving an *expected* running time of $O(\sqrt{n})$. In their original version, these algorithms have been proposed for $GF(p)^*$ [22].

The main idea here for computing x is to produce iteratively a sequence of elements $(d_i)_{i \geq 1}, d_i \in G$, where all the d_i's are of the form

$$a^k b^l.$$

x	Shanks	BJT
371 239 423	80 sec	55 sec
742 478 843	91 sec	80 sec
1 113 718 263	106 sec	101 sec
1 484 957 683	113 sec	128 sec
1 856 197 103	106 sec	105 sec

Table 1. Original Shanks and its Refinement (BJT)

So (d_i) will become periodic after at most n iterations. For a sequence producing group elements at random, this can be expected to happen in an expected number of $O(\sqrt{n})$ steps (birthday paradoxon). The computation stops when $d_{i'} \equiv d_i \bmod p$ for some pair (i, i') is recognized. Then

$$a^k b^l = d_i = d_{i'} = a^{k'} b^{l'}$$

and

$$a^{k-k'} b^{l-l'} = 1$$

and therefore

$$(k - k' + x(l - l')) \equiv 0 \bmod n,$$

which reveals x, provided that $\gcd(l - l', n) = 1$.

There are two methods of finding repetitions in sequences. Pollard's method finds the pair (i, i') by computing the sequence twice as (d_j) and (d_{2j}), waiting for $(d_j) = (d_{2j})$. This will be the case when j is a positive multiple of the period length.

Instead of computing the sequence twice, Brent's algorithm [3] remembers the sequence elements d_{2^i} and compares them to d_{2^i+j}, $1 \leq j \leq 2^i$. It can be shown that comparison is not needed for $1 \leq j \leq 3 \cdot 2^{i-1}$.

It is worthwhile examining different variants of producing the (d_i), because ideally the (d_i) should behave like a random sequence. In this case the expected length of this sequence can be shown to be $1.25\sqrt{n}$ until an element of G occurs twice. The original sequence of Pollard uses a partition of $G = S_1 \cup S_2 \cup S_3$ with equally sized S_i and is defined by

$$d_{i+1} := \begin{cases} ad_i, \; d_i \in S_1 \\ d_i^2, \; d_i \in S_2 \\ bd_i, \; d_i \in S_3 \end{cases}$$

where an arbitrary product of the form $a^k b^l$ may be chosen as a start value. The sequence of exponents of a produced by this definition are computed by $e_{i+1} = e_i + 1$ and $e_{i+1} = 2e_i$, starting from some $e_0 = k + xl$. A theoretical result about the "randomness" of that sequence seems not to be known. Teske [31] constructs a sequence where a distribution which is close to uniform can be proven. Her construction partitions G into 20 subsets $S_1, \ldots, S_{20}$ and replaces the definition of d_{i+1} above by

$$d_{i+1} := \begin{cases} m_k d_i, & d_i \in S_k \text{ and } 1 \leq k \leq 16 \\ d_i^2, & d_i \in S_k \text{ and } 17 \leq k \leq 20, \end{cases}$$

where the m_k are initially set to random power products of a and b.

By experiment she determines an average sequence length of $1.596\sqrt{|G|}$ for Pollard's original sequence and $1.292\sqrt{|G|}$ for her newly constructed sequence. The impact on actual computations is illustrated at prime order subgroups of elliptic curves over prime fields. For a subgroup size of 13 decimal digits the average running time over 40 runs of Pollard's original algorithm has been 27.3 minutes in contrast to 22.5 minutes of the improved version. This reflects the stable improvement of about 20% of the total running time observed for all group sizes.

A version which allows for parallel computations has been proposed in [32] and is of practical significance as long as there is no subexponential algorithm available for the group under consideration.

2.5 Shoup's Lower Bound

A recent result of Shoup shows that the two algorithms discussed above are optimal for groups where the group operations themselves are the only computations possible within an algorithm [29]. In fact, there are groups where no algorithms with running time better than $O(\sqrt{n})$ are known (elliptic curves, Jacobians of hyperelliptic curves).

Let G be a cyclic group of order n. Starting from the notion of an oracle which can be asked for the result of the group operations defined at the beginning of section 2, Shoup computes the probability that an algorithm outputs the correct answer to a DL problem in G after m oracle calls. The enumeration of elements of G can be thought of as encodings of the n (distinct) powers of a which are given by a map $\sigma : \mathbb{Z}/n\mathbb{Z} \longrightarrow S$, where S is a set of binary strings representing elements of G uniquely. The problem $a^x = b$ in G is then rewritten as follows. Given $(\sigma(1), \sigma(x))$, find a $y \in \mathbb{Z}/n\mathbb{Z}$ such that $\sigma(y) = \sigma(x)$.

We cite Shoup's main result concerning generic DL algorithms.

Theorem. *Let n be a positive integer whose largest prime divisor is p. Let $S \subset \{0,1\}^*$ be a set of cardinality at least n. Let A be a generic algorithm for $\mathbb{Z}/n\mathbb{Z}$ on S that makes m oracle queries, and suppose that the encoding function σ of $\mathbb{Z}/n\mathbb{Z}$ on S is chosen randomly. The input to A is $(\sigma(1), \sigma(x))$, where $x \in \mathbb{Z}/n\mathbb{Z}$ is random. The output of A is $y \in \mathbb{Z}/n\mathbb{Z}$. Then the probability that $x = y$ is $O(m^2/p)$.*

The conclusion is that for achieving a non-negligible probability for the event of success, one needs $O(\sqrt{p})$ oracle calls; thus a generic DL algorithm performs at least $O(\sqrt{p})$ group operations.

2.6 General Index Calculus

The *index calculus* methods which typically achieve a sub–exponential running time depend on the ability to efficiently generating so called *relations* among group elements. More precisely, if one fixes a (small) subset $S := \{g_1, \ldots, g_k\} \subset G$ and can find elements of the set

$$L := \{(e_1, \ldots, e_k) \mid g_1^{e_1} \cdots g_k^{e_k} = 1\}$$

in an efficient manner, then Shoup's lower bound is not cryptographically relevant for G. We note that L is a lattice in $\mathbb{Z}^k$ and

$$\begin{array}{rccc} \Phi: & \mathbb{Z}^k & \longrightarrow & G \\ & (e_1, \ldots, e_k) & \mapsto & g_1^{e_1} \cdots g_k^{e_k} \end{array}$$

a homomorphism with kernel L so that

$$\mathbb{Z}^k / L \cong G.$$

From the conditions of section 2.5 we see that the source of producing relations must use some information "outside" G. For example in $GF(p)^*$, we may use the fact that $GF(p)$ is a field, or in class groups of quadratic number fields, we have a reduction theory. Whenever G happens to have an environment with such properties, one can apply the index calculus method, an outline of which is given below.

1. choose factor base:
   fix a set $S := \{g_1, \ldots, g_k\} \subset G$ of group elements, where $g_1 := a$, $g_2 := b$
2. produce relations:
   find relations of the form
   $$g_1^{e_{i1}} \cdots g_k^{e_{ik}} = 1, \quad (1 \le i \le l)$$
   terminate this step when $\mathrm{rank}(e_{ij}) = k - 1$
3. linear algebra step:
   set $A := (e_{ij})$ and compute a solution to $Ax \equiv 0 \bmod |G|$
4. extract solution:
   Let $x = (x_1, \ldots, x_k)$ be a solution of step 3. Since the rank is $k - 1$, we must have $\lambda \cdot x \equiv (\log g_1, \ldots, \log g_k) \bmod |G|$ for some $\lambda \in \mathbb{Z}$. The logarithms with respect to g_1 – in case they all exist – are then found by setting $\lambda \equiv x_1^{-1} \bmod |G|$.

When analyzing an index calculus variant, several problems have to be addressed. The parameter k is subject to optimization. If k is too small, the time to find relations is probably too large; on the other hand, if k is too big, the linear algebra step consumes too much time. For step 2, the probability to find a relation must be taken into account. In step 3 the linear dependency is found modulo prime divisors of $|G|$, sparse matrix techniques such as Lanczos, Conjugate Gradient can be used. In this case, the running time of this step is $O(k^2 + k\omega)$, where ω is the total number of non–zero entries among the e_{ij}.

3 Elliptic Curves

Today, the best general discrete log algorithms for the group of points of an elliptic curve over a finite field K are the generic ones given in the preceding section. For two special cases, a more efficient way has been found so far. An efficient way to find relations in this group has not been found yet, so the index calculus idea is not applicable. Thus we are left with Shanks's and Pollard's algorithm, but due to the enormous space requirements of the former; only the latter one is actually applicable for larger groups in practice (say $|G| > 10^{15}$).

Let $f(X) = X^3 + a_2X^2 + a_1X + a_0 \in K[X]$.

An elliptic curve (EC) over K is defined as the following set of points

$$E = \{(x, y) \mid y^2 = f(x)\} \subset K \times K\} \cup \{\infty\}$$

if $\mathrm{char}(K) \neq 2$. By change of variables this may be transformed to

$$E = \{(x, y) \mid y^2 = x^3 + ax + b\} \subset K \times K\} \cup \{\infty\}$$

for appropriate $a, b \in K$ if $\mathrm{char}(K) \neq 3$.

If $\mathrm{char}(K) = 2$, an elliptic curve is given by

$$E = \{(x, y) \mid y^2 + y = f(x)\} \subset K \times K\} \cup \{\infty\}.$$

The addition law of two points $(x_1, y_1) + (x_2, y_2) = (x_3, y_3) \in E$ is given by the following rule ($\mathrm{char}(K) \notin \{2, 3\}$):

1. if $x_1 \neq x_2$ let $m := (y_2 - y_1)/(x_2 - x_1)$, and $m := (3x_1^2 + a)/2y_1$ otherwise
2. $x_3 := m^2 - x_1 - x_2$
3. $y_3 := -y_1 + m(x_1 - x_3)$.

For $\mathrm{char}(K) \in \{2, 3\}$ similar addition rules hold. With this addition, $(E, +)$ is an abelian group. Let K consist of q elements. By a theorem of Hasse, the number of elements of E is bounded by

$$q + 1 - 2\sqrt{q} \leq \#E \leq q + 1 + 2\sqrt{q}.$$

If $\#E = q + 1 - t$, we call t the *trace* of E.

3.1 General Elliptic Curves

Since 1997, there have been existing several public EC–DL challenges for finite fields [7]. By utilizing the ideas of [32] some of them have already been broken (electronic messages on the number theory net, Harley et al.), see table 2 on the following page.

$\#K$	# group operations
2^{79}	$1.7 \cdot 10^9$
p_{79}	$1.4 \cdot 10^9$
2^{89}	$1.8 \cdot 10^{13}$
p_{89}	$3.0 \cdot 10^{13}$
2^{97}	$2.2 \cdot 10^{13}$
p_{97}	$2.0 \cdot 10^{14}$

Table 2. Solved Elliptic Curve DL Challenges

Here, p_{79}, p_{89}, and p_{97} are 79–, 89–, and 97–bit primes respectively.
We now turn to the two special cases which can be solved efficiently.

3.2 Elliptic Curves of Trace Zero

In 1991, Menezes, Okamoto and Vanstone [21] published a method (MOV–reduction) to reduce the discrete log problem on E over $\mathbf{F}_q$ to the discrete log problem in $(\mathbf{F}_{q^k})^*$ for small k, provided that E is supersingular. A curve of $\mathbf{F}_q$ with $q = p^n$ is called supersingular if its trace is divisible by p. In particular this covers curves over $\mathbf{F}_p$ of order $p+1$. Consequently, this leads to a subexponential discrete log algorithm for supersingular curves.

Koblitz and Balusubramanian, however, showed that it is extremely unlikely for a random curve to be vulnerable by the MOV–reduction [2].

3.3 Elliptic Curves of Trace One

In 1997, Smart [30], and independently Semaev [26], Satoh and Araki [23] found an efficient method for curves E over $\mathbf{F}_p$, if E has p elements. The idea makes use of considering E over $\mathbb{Q}_p$, the p–adic extension of the rationals. In a certain related group to $E(\mathbb{Q}_p)$, there exists a logarithmic map which can be evaluated in polynomial time. It turns out that it suffices to approximate the arithmetic operations in $\mathbb{Q}_p$ by operations in $\mathbb{Z}/p^2\mathbb{Z}$.

4 Prime Fields

We now turn to a special version of the index calculus algorithm of section 2.6.

4.1 Sketch of the Number Field Sieve

Today the Number Field Sieve (NFS) is the asymptotically fastest known method to compute discrete logs in prime fields [14, 24]. Its running time is given by $L_p[1/3, (64/9)^{(1/3)}]$, where

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

This is the same running time as for factoring integers as large as p. This algorithm has been implemented and lead to a record of 85 decimal digits for general p as well as 129 decimal digits for special p. In this algorithm, the computations take place in two number rings $\mathbb{Z}[\alpha_1]$ and $\mathbb{Z}[\alpha_2]$, where α_i are zeros of two polynomials $f_i \in \mathbb{Z}[X]$, $i = 1, 2$ respectively. The number rings are linked to $(\mathbb{Z}/p\mathbb{Z})^*$ by two homomorphisms

$$\varphi_i : \mathbb{Z}[\alpha_i] \longrightarrow \mathbb{Z}/p\mathbb{Z}$$

with

$$\varphi(\alpha_1) = \varphi(\alpha_2).$$

Consequently, the factor bases consist of prime ideals of the ring of integers of $\mathbb{Q}(\alpha_i)$. Note that the factor base members are not element of the group in which the DL has to be computed. An early special case of the NFS, the Gaussian Integer method, was published in [10] which we obtain by setting $\mathbb{Z}[\alpha_1] = \mathbb{Z}$ and $\mathbb{Z}[\alpha_2]$ to be an imaginary quadratic principal ideal ring.

The relations consist of (small) pairs $(c, d) \in \mathbb{Z} \times \mathbb{Z}$, where the ideals $(c + d\alpha_1)$ and $(c + d\alpha_2)$ simultaneously split over the corresponding factor bases. The original number field sieve adaptation uses $\mathbb{Z}[\alpha_1] = \mathbb{Z}$ such that the members of the first factor bases which are (principal) prime ideals of $\mathbb{Z}$ can be interpreted as elements of G (take a generator r of each prime ideal and consider $\varphi_1(r) \in G$). In a generalization of that adaptation where $\mathbb{Z}[\alpha_1] \neq \mathbb{Z} \neq \mathbb{Z}[\alpha_2]$ we can compute the logarithm of $b \in G$ if there exists $\gamma \in \varphi_j^{-1}(b) \in \mathbb{Z}[\alpha_j]$ such that (γ) splits over factor base j for $j = 1$ or $j = 2$ [33]. We observe that the parameters of the DL problem, a and b, have to be small in order to show up as factors of elements of type $c + d\alpha_1$ and $c + d\alpha_2$. For the base a this is not a severe problem because a small generator mod p can be found in polynomial time [28]. The parameter b is usually reduced by finding an expression $b \equiv \prod s_i \bmod p$ with small s_i (see section 4.2).

As above a (sparse) matrix consisting of exponents (e_{ij}) is constructed, (e_{ij}) being the exponent of factor base element number j in relation number i. Reducing to prime order groups leads us to solving the linear algebra problem over $\mathbb{Z}/q\mathbb{Z}$ where $q|p-1$. Let $(y_3, \ldots, y_k)$ be a solution to $A'y = 0 \bmod q$ where the matrix A' consists of the rows $3, \ldots, k$ of A. Let (c_l, d_l) be the pair which produces relation number l $(1 \leq l \leq k)$.

Then

$$\prod \varphi_1(c_l + d_l\alpha_1)^{y_l} = g_1^{x_1} g_2^{x_2} u^q$$
$$\prod \varphi_2(c_l + d_l\alpha_2)^{y_l} = v^q$$

for some $u, v \in \mathbb{Z}/p\mathbb{Z}$ and $x_1, x_2 \in \mathbb{Z}$, provided that two conditions hold. Firstly, the ideals $\prod \varphi_1(c_l + d_l\alpha_j)^{y_l}$ have to be q–th powers of principal ideals; secondly, the units which are congruent to 1 mod q must be q–th powers. These assumptions, though heuristic, can be reasonably justified, see [24].

Because of $\varphi_1(c_l + d_l\alpha_1) = \varphi_2(c_l + d_l\alpha_2)$, dividing both equations results in a DL solution modulo q:

$$g_1^{x_1} g_2^{x_2} \equiv (v/u)^q \bmod p.$$

In practice, numerous refinements have been discovered during the few years after the first implementation of the NFS [6]. The most effective one surely is the large prime variation, where relations of the form

$$g_1^{e_1} \cdots g_k^{e_k} \cdot h = 1$$

are also accepted, where h (the large prime) is not member of the factor base. Dividing a second relation of this form by the one given above immediately produces an ordinary relation. This has been extended to allowing multiple h's, usually two per factor base. Limiting the number of h's is due to the method they are recognized. After the sieving process, all powers of factor base elements are found. Therefore, allowing one h requires one primality test per relation after the sieving stage. Allowing a rest of the form $h_1 \cdot h_2$ already requires the use of a fast special purpose factoring method to extract h_1 and h_2. This is practical as long as the h's fit in a computer word, so our special purpose factoring method has to split integers of size 2^{64}. In literature, the use of Pollard's $p-1$ method, Shanks's square form factorization method and Lenstra's elliptic curve method are reported. The combination of relations having more than one large prime is non–trivial. The basic case of two large primes in factoring algorithms was settled by [19], extended by [37] to four large primes, and adapted to the discrete log case in [34].

The distribution of the sieving process on many workstations and running the linear algebra step on a massively parallel machine (usually the Lanczos method or the conjugate gradient method) are subject of optimization, too [12, 17, 18].

For general finite fields $GF(p^n)$, the number field sieve ($n < (\log p)^{1/2}$) and the function field sieve ($n > (\log p)^2$) can be employed. For the "gap" between both of them, we are not aware of a subexponential time algorithm. A detailed theoretical survey of both methods and on the current status concerning the gap can be found in [25].

4.2 Experimental Results

Perhaps the most remarkable result of the DL variant of the NFS has probably been the successful solving of McCurley's 129–digit discrete logarithm challenge [36], which McCurley published in his overview paper on the DL problem [20].

In view of the Diffie–Hellman key exchange protocol introduced in [13], McCurley stated a challenge by using the following setup:

$$\begin{aligned} b_A &= 127402180119973946824269244334322849749382042586931621654557735290322914679095998681860978813046595166455458144280588076766033781 \\ b_B &= 180162285287453102444782834836799895015967046695346697313025121734059953772058475958176910625380692101651848662362137934026803049 \\ p &= (739 \cdot 7^{149} - 736)/3 \\ q &= (p-1)/(2 \cdot 739). \end{aligned}$$

The order of the multiplicative group, which is generated by the element 7, splits as follows: $|(\mathbb{Z}/p\mathbb{Z})^*| = 2 \cdot 739 \cdot q$.

- Alice computes (using her secret key x_A) as $7^{x_A} \equiv b_A \pmod{p}$.
- Bob computes (using his secret key x_B) as $7^{x_B} \equiv b_B \pmod{p}$.

Kevin McCurley asked for the common secret key $K \equiv 7^{(x_A \cdot x_B)} \pmod{p}$ which has been computed by Denny and the second author as

$$K = 38127280411190014138078391507929634193998643551018670285056375615045523966929403922102172514053270928872639426370063532797740808, \tag{3}$$

by first calculating

$$x_A = 618586908596518832735933316520379042679876430695217134591462221849525998156144877820757492182909777408338791850457946749734, \tag{4}$$

the secret key of Alice.

Since the probability of $c + d\alpha$ splitting over a factor base depends on the discriminant of $\mathbb{Z}[\alpha]$, primes of special form – such as McCurley's choice of p – serve as an attractive target for the NFS.

Within a total of $\approx$ 180 mips years (mips=mega instructions per second), enough large prime relations (with a maximum of two large primes per relation) have been found over two factor bases of 35000 elements each.

Within a total of 1200 CPU hours, distributed on Sparc 4 workstations, the following reduction of b has been found by a combination of trial division and the elliptic curve factoring method:

$$a^{141266132} \cdot b \equiv \frac{t}{v} \pmod{p},$$

where

$$\begin{aligned} t &= 2^3 \cdot 31 \cdot s_1 \cdot s_3 \cdot s_6 \cdot s_8 \cdot s_{10} \cdot s_{11} \\ v &= 353 \cdot s_2 \cdot s_4 \cdot s_5 \cdot s_7 \cdot s_9 \cdot s_{12}. \end{aligned} \tag{5}$$

with

$$\begin{aligned}
s_1 &= 603623,\\
s_2 &= 165073039,\\
s_3 &= 1571562367,\\
s_4 &= 1601141623,\\
s_5 &= 1715568391,\\
s_6 &= 7575446399,\\
s_7 &= 13166825869,\\
s_8 &= 265542836371,\\
s_9 &= 371303006453,\\
s_{10} &= 4145488613977,\\
s_{11} &= 4338202139093,\\
s_{12} &= 5041332876473.
\end{aligned}$$

With the aid of 15 solutions of the linear algebra problem, we are able to determine the log of the 15 elements 2, 31, 353 and $s_1, \ldots, s_{12}$. This step costs 911 hours on a Sparc 20 workstation and consumes 30 MB of main memory.

Alternatively, reducing the DL problem to smaller elements can be achieved by means of a recent sieving method [35].

For primes of arbitrary form, the latest official record of a 85–digit p [35] has been superseded by a 90–digit computation in May 1998 (electronic message on the number theory net, Gaussian Integer implementation by Lercier/Joux). In the 85–digit computation, the Gaussian Integer method has been compared to the NFS version with two quadratic polynomials (NFS2Q). With 30 mips years, only 2/3 of the NFS2Q time is needed, and also the time for computing the linear algebra solution has only been 1/3 of the NFS2Q version. The main reason for both observations is that the numbers, which have to be split during the algorithm, are slightly harder to factor over an equally sized factor base. So, more pairs (c, d) have to be tested during the sieving stage, and, during the elimination of large primes, more partials are needed to produce an ordinary relation. The latter obviously increases the number of non–zero entries in the relation matrix.

5 Conclusion

During the past few years there has been considerable progress on the ability to solve discrete logarithm problems. It is a matter of taste whether schemes are considered as secure, when subexponential attacks exist. As sharp lower complexity bounds are typically hard to achieve, it is possible that exponential attacks can be replaced by subexponential attacks, and subexponential attacks by polynomial time attacks. The DL problem, however, will serve as a reliable source for secure protocols as long as the cryptographers are not running out of appropriate groups where no subexponential time algorithm is known. But there is still the possibility that eventually all DL problems will be tackled easily.

We draw the conclusion that apart from DL and factoring, modern cryptography urgently needs further number theoretic problems which can safely be used as a setting for existing and future cryptographic protocols. For achieving provable security, lower bounds in solving the corresponding problems are essential but these are either too difficult to establish or too restricted to a model not sufficiently compliant with reality as in section 2.

References

1. L. M. Adleman, J. DeMarrais, and M.-D. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In *Algorithmic number theory*, number 877 in Lecture Notes in Computer Science, pages 28–40, 1994.
2. R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes–Okamoto–Vanstone algorithm. *Journal of Cryptology*, 11:141–145, 1998.
3. R. P. Brent. An improved monte carlo factorization algorithm. *Nordisk Tidskrift for Informationsbehandling (BIT) 20*, pages 176–184, 1980.
4. J. Buchmann and St. Düllmann. On the computation of discrete logarithms in class groups. In *Advances in Cryptology – Crypto '90*, number 537 in Lecture Notes in Computer Science, pages 134–139, 1991.
5. J. Buchmann, M. Jacobson, and E. Teske. On some computational problems in finite abelian groups. *Math. Comp.*, 66(220):1663–1687, 1987.
6. J. Buchmann, J. Loho, and J. Zayer. An implementation of the general number field sieve. In *Advances in Cryptology – Crypto '93*, number 773 in Lecture Notes in Computer Science, 1993.
7. Certicom. ECC challenge. `http://www.certicom.com/chal/`, 1997.
8. D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology – Eurocrypt'87*, number 304 in Lecture Notes in Computer Science, pages pp. 127–141, 1988.
9. D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In *Advances in Cryptology – CRYPTO'86*, number 263 in Lecture Notes in Computer Science, pages pp. 200–212, 1987.
10. D. Coppersmith, A. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica 1*, pages 1–15, 1986.
11. Th. Corman, Ch. Leiserson, and R. Rivest. *Introduction to algorithms.* MIT Press/McGraw–Hill, 1990.
12. Th. F. Denny. *Lösen grosser dünnbesetzter Gleichungssysteme über endlichen Primkörpern.* PhD thesis, Universität des Saarlandes/Germany, 1997.
13. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Information Theory 22*, pages pp. 472–492, 1976.
14. D. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math.*, 6:124–138, 1993.
15. N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48:203–209, 1987.
16. N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.
17. M. LaMacchia and A. Odlyzko. Solving large sparse linear systems over finite fields. In *Advances in Cryptology – Crypto '90*, number 537 in Lecture Notes in Computer Science, pages 109–133, 1990.

18. R. Lambert. *Computational aspects of discrete logarithms.* PhD thesis, University of Waterloo/Canada, 1996.
19. A. K. Lenstra and M.S. Manasse. Factoring with two large primes. *Math. Comp.*, 63:77–82, 1994.
20. K. S. McCurley. The discrete logarithm problem. In *Cryptology and Computational Number Theory*, number 42 in Proc. Symp. in Applied Mathematics, pages 49–74. American Mathematical Society, 1990.
21. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 80–89, 1991.
22. J. M. Pollard. Monte carlo methods for index computation $(\bmod p)$. *Math. Comp.*, 32:918–924, 1978.
23. T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. preprint.
24. O. Schirokauer. Discrete logarithms and local units. *Phil. Trans. R. Soc. Lond. A 345*, pages 409–423, 1993.
25. 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.
26. I. A. Semaev. Evaluation of descrete logarithms on some elliptic curves. *Math. Comp.*, 67:353–356, 1998.
27. D. Shanks. Class number, a theory of factorization and genera. In *Proc. Symposium Pure Mathematics*, volume 20, pages 415–440. American Mathematical Society, 1970.
28. V. Shoup. Searching for primitive roots in finite fields. In *Proc. 22nd Annual ACM Symp. on Theory of Computing (STOC)*, pages 546–554, 1990.
29. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in cryptology – Eurocrypt'97*, number 1233 in Lecture Notes in Computer Science, pages 256–266, 1997.
30. N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology.* to appear.
31. E. Teske. Speeding up pollard's rho method for computing discrete logarithms. In *Algorithmic Number Theory – ANTS III*, number 1423 in Lecture Notes in Computer Science, 1998.
32. P. van Oorschot and M. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology.* to appear.
33. 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.
34. D. Weber. *On the computation of discrete logarithms in finite prime fields.* PhD thesis, Universität des Saarlandes/Germany, 1997.
35. D. Weber. Computing discrete logarithms with quadratic number rings. In *Eurocrypt'98*, number 1403 in Lecture Notes in Computer Science, 1998.
36. D. Weber and Th. Denny. The solution of McCurley's discrete log challenge. In *Advances in Cryptology – CRYPTO'98*, number 1462 in Lecture Notes in Computer Science, 1998.
37. J. Zayer. *Faktorisieren mit dem Number Field Sieve.* PhD thesis, Universität des Saarlandes/Germany, 1995.