Discrete Logarithms and the Effectiveness of the Index Calculus Method

Oliver Schirokauer[1], Damian Weber[2], Thomas Denny[2]

[1] Department of Mathematics
Oberlin College
Oberlin
Ohio 44074
U.S.A.
e-mail: `oliver@pekochan.math.oberlin.edu`

[2] Universität des Saarlandes
FB 14 Informatik
Postfach 15 11 50
66041 Saarbrücken
Germany
e-mail: `dweber@cs.uni-sb.de, denny@cs.uni-sb.de`

Abstract. In this article we survey recent developments concerning the discrete logarithm problem. Both theoretical and practical results are discussed. We emphasize the case of finite fields, and in particular, recent modifications of the index calculus method, including the number field sieve and the function field sieve. We also provide a sketch of the some of the cryptographic schemes whose security depends on the intractibility of the discrete logarithm problem.

1 Introduction

Let G be a cyclic group generated by an element t. The discrete logarithm problem in G is to compute for any $b \in G$ the least non-negative integer e such that $t^e = b$. In this case, we write $\log_t b = e$. Our purpose, in this paper, is to survey recent work on the discrete logarithm problem. Our approach is twofold. On the one hand, we consider the problem from a purely theoretical perspective. Indeed, the algorithms that have been developed to solve it not only explore the fundamental nature of one of the basic structures of mathematics but also are themselves intricate, abstract objects which incorporate results and raise questions from a variety of mathematical areas. On the other hand, we recognize the pleasure of a successful implementation and the growing cryptographic interest in actual computations. We therefore present computational results whenever available and give a brief discussion in §2 of some practical cryptographic schemes.

At first glance, the discrete logarithm problem as described above does not seem difficult. If G is the additive group $\mathbb{Z}/n\mathbb{Z}$ represented by the integers

$\{0, \ldots, n-1\}$ and we let $t = 1$, then it is a trivial matter to compute $\log_t b$. The groups we consider in this paper, however, are represented in such a way that the cyclic structure is hidden. In other words, the isomorphism that exists between G and $\mathbb{Z}/n\mathbb{Z}$ is not apparent. In some cases the representation has properties that enable us, after much labor, to uncover the group structure. In §3 and §4, for instance, we consider the multiplicative group of a finite field. We represent the finite field either as the quotient of a polynomial ring over a prime field or the quotient of a number ring. The notion of smoothness in both types of rings allows us to use an algorithm known as the index calculus method. In the case that G is the class group of a number field, we will see in §5 that a variation of the index calculus method can be used. In contrast, when G is the group of points on an elliptic curve, no version of the index calculus method has been developed, a fact which makes these groups particularly attractive for cryptographic purposes.

The discrete logarithm problem is often compared to the problem of factoring an integer. Many of the most prominent modern factoring algorithms have a discrete logarithm analogue. The heuristically fastest factoring algorithm, the number field sieve (NFS), is no exception. It has a conjectured expected running time of

$$L_n[1/3; c + o(1)] \quad \text{for} \quad n \to \infty,$$

where n is the number being factored, c is a constant, and

$$L_n[s; c] = \exp(c\,(\log^n)^s (\log\log n)^{1-s}).$$

In this case, there are two corresponding discrete logarithm algorithms, the number field sieve and the function field sieve. Both of these algorithms apply to the case that G is the multiplicative group of a finite field and both have, for a broad range of finite fields, a conjectured expected running time equal to the NFS factoring algorithm, with n replaced by the cardinality of the finite field. Together, however, they do not achieve this running time for all fields and it remains an open question to find an algorithm as fast as the NFS for all finite fields. In practice the NFS has been implemented for prime fields, and the FFS, in the form of Coppersmith's algorithm, for fields of characteristic 2. Despite the success of these methods, the computation of discrete logarithms in finite fields lags behind the factoring of integers of size comparable to that of the finite field. In part, this is simply due to the fact that less attention has been paid to the discrete logarithm problem. In part, it reflects the presence of obstacles that are particular to the discrete logarithm problem. One such impediment is the linear algebra, which we discuss at the end of §4. In §6 we give a more general account of open questions, both practical and theoretical.

2 Cryptography

We briefly describe three of the many cryptographic protocols whose security depends on the difficulty of computing discrete logarithms and which are of practical use to gain secret communication. Among those we do not discuss are identification schemes ([60]) and hash functions ([16]). Note that, though our second and third examples are formulated for the case that G is the multiplicative group of a finite field, these schemes can be modified for other groups G, including the group of points of an elliptic curve over a finite field. For a more extensive discussion of elliptic curve cryptosystems, we refer the reader to [31], [47],[45].

Throughout this section, we will use the expression $x \text{ Mod } y$ to denote the least positive residue of x modulo y.

2.1 The Diffie–Hellman protocol

A simple cryptographic scheme that uses the difficulty of the discrete logarithm problem is the Diffie–Hellman key exchange protocol, which works for all groups [22]. Suppose A and B are two persons who want to agree upon a secret key, but only have an unsafe communication channel. First they choose a group G and an element $a \in G$. For security purposes, the order of a should have at least one large prime factor. Next, A picks secretly $x_{\mathrm{A}} \in \mathbb{Z}$ and likewise B picks $x_{\mathrm{b}} \in \mathbb{Z}$. Then A sends $a^{x_{\mathrm{A}}}$ to B and B sends $a^{x_{\mathrm{B}}}$ to A. Both A and B can compute $a^{x_{\mathrm{A}}x_{\mathrm{B}}}$, which is their secret key. It follows at once that if the discrete logarithm problem can be solved in G, then the Diffie–Hellman protocol can be broken. There is a current challenge of McCurley which requires breaking the Diffie–Hellman protocol [43]. He chooses for his group the multiplicative group of $\mathbb{Z}/p\mathbb{Z}$, where p=$(739 \cdot 7^{149} - 1472)/3$. The decimal representation of p has 129 digits. For details concerning a first step in attacking the challenge see §4.3 and [68].

The Diffie–Hellman problem (DH) for a cyclic group G with generator g is, given g^a, g^b for $a, b \in \mathbb{Z}$, to compute the element $g^{ab} \in G$. Boneh and Lipton in [9] show that if a subexponential algorithm for DH in a group G exists, then a subexponential algorithm for the discrete logarithm problem (DL) exists. In what follows, however, we concern ourselves with polynomial equivalence. To make this precise, we introduce a DH–oracle for $G = \langle g \rangle$, which on input $(g^a, g^b) \in G \times G$ outputs $g^{ab} \in G$ within constant time. The question then is whether it is possible to compute x from g^x by using polynomially many operations in G and polynomially many calls to the DH–oracle for G. In general, the question is unanswered.

In [8], den Boer demonstrated the polynomial equivalence of DH and DL in $(\mathbb{Z}/p\mathbb{Z})^*$, for p prime, in the case that the totient of $p-1$ is smooth with respect to a bound which is polynomial in $\log p$. Maurer ([42]) extended the set of groups for which the equivalence holds to include groups G such that for all p dividing $|G|$ and greater than some bound B which is polynomial in $\log |G|$, either $p-1$ or $p+1$ is smooth with respect to B. Subsequently, Maurer and Wolf ([44]) further extended the set to include those G such that for all p dividing $|G|$ and greater than some bound B which is polynomial in $\log |G|$, either $p-1$ or $p+1$ is smooth or $\phi_n(p)$ is smooth for some n, where ϕ_n is the nth cyclotomic polynomial and n is bounded.

The idea introduced by Maurer and then used by Maurer and Wolf is to transport the discrete logarithm problem in a goup G to an auxiliary group H which has smooth order but which requires the DH–oracle in order to perform multiplication. The groups H that are introduced are one of two types, either the group of points of an elliptic curve over a finite field or the subgroup of the multiplicative group of a finite field. The equivalence is obtained by now using a discrete logarithm algorithm in H, known as the Silver-Pohlig-Hellman method (see [43]), which takes advantage of the smoothness of $|H|$.

2.2 Public key

The idea of a public key cryptosystem is that anybody is able to send an encrypted message m to a participant A by using a publicly known encryption key e_{A} while only A can decrypt m with the aid of his or her secret private decryption key d_{A}. For the sake of security against opponents, it must be computationally infeasible to derive d_{A} from e_{A}.

The idea of public key cryptography first appeared in an article of Diffie and Hellman [22] in 1976. Soon thereafter, Rivest, Shamir, and Adleman invented the well–known RSA public-key cryptosystem [57], which depends for its security on the difficulty of the integer factorization problem. In 1985, ElGamal published a public key system based on the difficulty of the discrete logarithm problem in the multiplicative group of a prime field [25]. For the description, we follow [65].

Suppose p is a prime such that the discrete logarithm problem in $(\mathbb{Z}/p\mathbb{Z})^*$ is intractible, and let a be a generator of $\mathbb{Z}/p\mathbb{Z}^*$. To begin with, A chooses a secret key $d_{\mathrm{A}} \in \mathbb{Z}$ and publishes $a^{d_{\mathrm{A}}}$ Mod p, an integer we will call e_{A}. In order to encrypt a message m which is to be read by A, one chooses a random $k \in \mathbb{Z}/(p-1)\mathbb{Z}$ and computes the pair (y_1, y_2), where

$$y_1 = a^k \text{ Mod } p \qquad y_2 = me_{\mathrm{A}}^k \text{ Mod } p.$$

Now since

$$\frac{y_2}{y_1^{d_{\mathrm{A}}}} \equiv \frac{me_{\mathrm{A}}^k}{a^{kd_{\mathrm{A}}}} \equiv \frac{ma^{kd_{\mathrm{A}}}}{a^{kd_{\mathrm{A}}}} \equiv m \bmod p,$$

A can decrypt the message by evaluating the inverse of $y_1 \bmod p$ and then computing $y_2(y_1^{-1})^{d_{\mathrm{A}}}$ Mod p.

2.3 Digital signatures

An important part of cryptography is the development of secure signature schemes. The problem, in this case, is to find a way to sign a message so that the signature cannot be forged and can be verified by anyone. To prevent forging, the difficulty of the discrete logarithm problem can be used. For verification, discrete exponentiation (powering by using repeated squaring) can be employed. In December 1994, The National Institute of Standards and Technology adopted a modification of the ElGamal Signature Scheme [25] known as the Digital Signature Standard (DSS) [49]. We follow the description of the DSS in [65].

Let p be a 512–bit prime such that the discrete log problem in $(\mathbb{Z}/p\mathbb{Z})^*$ is intractible, q a 160–bit prime with $q|(p-1)$, and a an element of order q in $(\mathbb{Z}/p\mathbb{Z})^*$. Assume person A has a secret key d_{A}. Let $e_{\mathrm{A}} = a^{d_{\mathrm{A}}}$ Mod p. As in the previous scheme, A publishes e_{A}. Whenever signing a message m, A randomly determines a parameter $k \in (\mathbb{Z}/q\mathbb{Z})$ and computes a pair $(\gamma, \delta) \in \mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ according to

$$\gamma = (a^k \text{ Mod } p) \text{ Mod } q, \quad \delta = (m + d_{\mathrm{A}}\gamma)k^{-1} \text{ Mod } q.$$

This pair is then appended to the message. Note that a has order q mod p, so a^k mod p is well–defined. Anyone can now verify the signature (γ, δ) by checking whether

$$(a^{m\delta^{-1}} \cdot e_{\mathrm{A}}^{\gamma\delta^{-1}} \text{ Mod } p) \text{ Mod } q = \gamma.$$

3 Algorithms for finite fields

Let $q = p^n$, where p is a prime number and n a positive integer. Let $\mathbb{F}_q$ be the field of q elements and assume that t generates $\mathbb{F}_q{}^*$. Let b be a second element in $\mathbb{F}_q{}^*$. There are various algorithms for computing $\log_t b$ which have an exponential running time. These include Shank's baby step/giant step method, Pollard's rho method, and the Silver-Pohlig-Hellman method, all of which are described in [37]. Here we pause only to comment on the success of the last of these in solving logarithms in a prime field. With the practical improvements of Pollard [55] and Brent [10], the Silver-Pohlig-Hellman algorithm was used compute $\log_7 11$ in $\mathbb{F}_p$, where

$$p = 3108193814121911011419.$$

The solution

$$x = 2943721236853854327 16$$

was computed in 10718 CPU–minutes on a Sparc ELC workstation (approx. 7.5 days) [67]. The significance of this example lies in the fact that $(p-1)/2$ is a prime of 21 digits. The running time of the Silver-Pohlig-Hellman method for a field $\mathbb{F}_q$ depends on the largest prime factor of $q-1$. The algorithm, therefore, is most effective when the prime factors of $q-1$ are small. In this case, we see its practicality in a prime field even when the cardinality of the multiplicative group has a factor of 21 digits.

Most recent work on the discrete logarithm problem for finite fields has focused on analyzing, modifying, and implementing various forms of the index calculus method. For the remainder of this section, we concentrate on this approach. The reader will find a general description of the index calculus method in just about any survey article on discrete logarithms, including [37],[43],[51], and [52]. The reader will also find in these works the names of the many mathematicians who discovered this method, the earliest being Kraitchik who wrote about it in the 1920's ([33]). We begin with a brief description of the method in the simplest setting, a prime field, and then give a more abstract formulation which will lend itself to the various algorithms that are discussed subsequently.

Let p be a prime and assume we are interested in computing $\log_t b$ in $(\mathbb{Z}/p\mathbb{Z})^*$. We take as representatives for $\mathbb{Z}/p\mathbb{Z}$ the set $F = \{0, 1, \dots, p-1\}$. Let S be the set of primes in F which are less than or equal to some given bound. The first step of the algorithm is to find integers k such that t^k is represented by an element of F which factors over S. In this way we obtain relations in $\mathbb{Z}/p\mathbb{Z}$ of the sort

$$t^k = \prod_{s \in S} s^{e_s},$$

where s in this case denotes the coset in $\mathbb{Z}/p\mathbb{Z}$ represented by s. Each such equation yields a linear equation of the form

$$k = \sum_{s \in S} e_s \log_t s.$$

We also search for a single exponent c such that $t^c b$ factors over S. This relation gives an equation of the sort

$$c + \log_t b = \sum_{s \in S} e_s \log_t s.$$

Once we have accumulated enough relations so that the linear sytstem has full rank, we can use linear algebra mod $p-1$ to solve for the logarithms of all the primes in S and, more importantly, for $\log_t b$.

Now let R be a Dedekind domain with field of fractions K. Recall that every ideal in R factors uniquely into prime ideals. Assume that there exists a ring homomorphism $\phi : R \to \mathbb{F}_q$. Let S be a set of prime ideals of R and call an element of R smooth if the ideal it generates factors over S. The set S is called the factor base. Let τ and β be such that $\phi(\tau) = t$ and $\phi(\beta) = b$ and assume β is smooth. The first step in our algorithm is to find many pairs $(\delta, \gamma) \in R \times R$ such that

i) $\phi(\delta) = \phi(\gamma)$

ii) δ factors into a power of τ and a smooth element

iii) γ is smooth.

Assume the factorizations of (δ) and (γ) are given by

$$(\delta) = (\tau)^{v_0} \prod_{j=1}^{|S|} P_j^{v_j} \quad \text{and} \quad (\gamma) = \prod_{j=1}^{|S|} P_j^{w_j},$$

where the P_j are the prime ideals in S. Then for each pair, we obtain a vector $v_\delta = (v_j)$ where $j = 0, \dots, |S|$ and a vector $w_\gamma = (w_j)$ where this time $j = 1, \dots, |S|$. Additionally we define a vector $v_t = (1, 0, \dots, 0)$ and a vector w_b containing the exponents in the factorization

$$(\beta) = \prod_{j=1}^{|S|} P_j^{w_j}.$$

In the second stage of the algorithm we use linear algebra modulo $(q-1)$ to find integers $e_{(\delta,\gamma)}$ and a single integer e so that

$$ev_t + \sum e_{(\delta,\gamma)} v_\delta \equiv 0 \bmod (q-1) \quad \text{and} \quad w_b + \sum e_{(\delta,\gamma)} w_\gamma \equiv 0 \bmod (q-1).$$

Let $\mu = \tau^e \prod \delta^{e_{(\delta,\gamma)}}$ and let $\nu = \beta \prod \gamma^{e_{(\delta,\gamma)}}$. Then

$$b\phi(\mu) = bt^e \phi(\prod \delta^{e_{(\delta,\gamma)}}) = t^e b\phi(\prod \gamma^{e_{(\delta,\gamma)}}) = t^e \phi(\nu).$$

Furthermore, μ and ν are elements in the set

$$V = \{\alpha \in K^* \mid (q-1)|\mathrm{ord}_P \alpha \;\; \text{for all prime ideals } P \subset R\}.$$

Though V contains the set of $(q-1)$st powers in K^*, which we will denote by K^{*q-1}, it is not in general equal to K^{*q-1}. Assume for the moment, however, that μ and ν are $(q-1)$st powers. Since the $(q-1)$st power of any element in $\mathbb{F}_q$ is 1, we see immediately that $b = t^e$ and our problem is solved.

As is evident, one issue that arises in analyzing this algorithm is the discrepancy between V and K^{*q-1}. Indeed, the quotient V/K^{*q-1} can be thought of as an obstruction group. It is easily seen to be a $\mathbb{Z}/(q-1)\mathbb{Z}$ module, and for the R we consider below, it is finitely generated. When its only elements are the images of roots of unity in V, in which case its rank is 1, this obstruction group poses little difficulty. When it has larger rank, some more severe modifications are necessary.

3.1 Rigorous subexponential algorithms

Since a finite field $\mathbb{F}_q$ can be represented as either a quotient of $\mathbb{Z}$ or a quotient of $\mathbb{F}_p[X]$, these two rings make natural choices for R. Indeed, the map $\phi : R \to \mathbb{F}_q$ is then simply the quotient map. Since $\mathbb{Z}$ and $\mathbb{F}_p[X]$ are both principal ideal domains with few units, the index calculus method takes on a particularly simple form with only a small obstruction group. Moreover, results about smooth elements in these rings allow for a rigorous analysis of the algorithm.

We begin with the case that $R = \mathbb{F}_p[X]$ and assume that $\mathbb{F}_q$ is represented as $\mathbb{F}_p[X]/f(X)$, where $f(X)$ is an irreducible polynomial of degree n. The elements t and b, then, are cosets and we let τ and β be representatives for them of smallest degree. As stated, $\phi : R \to \mathbb{F}_q$ is the quotient map. Let S be the set of ideals generated by an irreducible polynomial of degree less than or equal to some bound B. Given an element $g(X) \in \mathbb{F}_q[X]$, we denote by $\overline{g(X)}$ the polynomial of smallest degree congruent to $g(X)$ modulo $f(X)$. The pairs then that are tested for smoothness in the first stage of the algorithm are $(\tau^k, \overline{\tau^k})$ for randomly chosen values of $k \in \{1, 2, \ldots, q-1\}$. Notice that the vectors v_δ in this case have only one non-zero entry. Since β is not necessarily smooth, an extra search is done to find s so that $\overline{\tau^s \beta}$ is smooth. In this case, we let the vector w_b contain the exponents occuring in the factorization of $(\tau^s \beta)$. As a result, the algorithm produces $\log_t t^s b$, from which $\log_t b$ can be deduced. Since $\mathbb{F}_p[X]$ is a principal ideal domain, the obstruction group $V/K^{*q-1} \cong \mathbb{F}_p[X]^*$. But the only units in $\mathbb{F}_p[X]$ are the non-zero elements of the base field. We conclude that our algorithm produces a relation of the sort $b = at^e$ for $a \in {\mathbb{F}_p}^*$. Since computing $\log_t a$ is an easy matter when p is small, our problem is solved in this case. In what follows we will refer to this algorithm as the IC-PR algorithm, the letters standing for Index Calculus - Polynomial Ring.

The running time of the IC-PR algorithm depends on the probability that a polynomial of degree $\leq n$ is B-smooth. In 1985, Odlyzko ([51]) determined this probability in the case that $p = 2$. As a result, Pomerance ([56]) was able to prove in 1987 that the algorithm, in the case that q is a power of 2, has an expected running time of $L_q[1/2; \sqrt{2} + o(1)]$, where the limit implicit in the $o(1)$ is for $q \to \infty$. Since then Odlyzko's arguments have been extended and modified. In 1992 Lovorn-Bender ([39]) used the same approach as Odlyzko to determine an asymptotic formula for the number of B-smooth polynomials of degree $\leq n$ in $\mathbb{F}_p[X]$, under the condition that $n^{1/100} \leq B \leq n^{99/100}$. With this in hand, she was able to prove that the IC-PR algorithm has an expected running time of $L_q[1/2; \sqrt{2} + o(1)]$, so long as $\log p \leq n^{.98}$ as $q \to \infty$. Finally, Lovorn-Bender and Pomerance ([41]) recently were able to extend this result and show that the algorithm has an expected running time of $L_q[1/2; \sqrt{2} + o(1)]$ so long as $p \leq n^{o(n)}$ for $q \to \infty$. Moreoever, they show that it is a subexponential algorithm under the far weaker assumption that $n \to \infty$ as $q \to \infty$. Their improvements depend on a result of Soundararajan ([64]) which gives the number of B-smooth polynomials of degree $\leq n$ for $2\log^2 n \leq B \leq n$ and a result of their own giving a lower bound for this quantity when $B \leq n^{1/2}$.

Despite the breadth of Lovorn-Bender and Pomerance's latest result, there are many cases for which the IC-PR algorithm is not subexponential. In particular, if n is fixed as $q \to \infty$ a different approach is necessary. For $n = 1$, it is best to let $R = \mathbb{Z}$. We do not give the details here as there have not been any recent developments in the analysis of this version of the index calculus method. See [56] for Pomerance's proof that this algorithm has an expected running time of $L_q[1/2; \sqrt{2} + o(1)]$.

In the case that $n = 2$, we encounter in the work of El Gamal ([26]) and subsequently Lovorn-Bender ([39],[40]), a new choice for R, namely the ring of integers of a quadratic extension of $\mathbb{Q}$. We briefly describe the version of Lovorn-Bender. Let R be the ring of integers of a quadratic imaginary field and assume that p is inert in R. In other words, $R/pR \cong \mathbb{F}_q$. Let $\phi : R \to \mathbb{F}_q$ be the quotient map and let S be the set of prime ideals whose norm is prime and less than or equal to some given bound. We find an integral basis $\{1, \alpha\}$ for R and fix $T = \{a + b\alpha \,|\, a, b \in \{0, \ldots, p-1\}\}$ as a set representatives for R/pR. For any $r \in R$ we denote by $\overline{r}$ the element in T congruent to r modulo p, and we pick τ and β from T so that $\phi(\tau) = t$ and $\phi(\beta) = b$. At this point, the algorithm proceeds in exactly the same way as the IC-PR algorithm, except that the linear algebra is done modulo $(q-1)^y$, where y is chosen so that

$$V' = \{\alpha \in K^* \,\big|\, (q-1)^y | \mathrm{ord}_P \alpha \;\; \text{for all prime ideals } P \subset R\}$$

satisfies $V'/K^{*q-1} \cong R^*$. It is not hard to show that such a y exists and to compute a bound for it. The obstruction group, then, only consists of R^*, which has cardinality at most 6 since R is an imaginary quadratic number ring. Such an obstruction is easily handled.

Lovorn-Bender is able to prove that her algorithm has an expected running time of $L_q[1/2; 3/2 + o(1)]$ for $p \to \infty$. To do so requires determining the probability that an element in T is smooth. In El Gamal's work, this analysis is done under the assumption of the Generalized Riemann Hypothesis. Lovorn-Bender is able to avoid the GRH by randomizing her choice of R. The method involves making a random choice from a small set of discriminants which are obtained by means of multipliers. The idea is also used to avoid the GRH in the factoring algorithm of [38].

3.2 Heuristic subexponential algorithms

In the last few years, various generalizations of the rigorous algorithms described above have appeared. In all of them R is either taken to be an extension of $\mathbb{F}_p[X]$ or an extension of $\mathbb{Z}$. What is gained is a broader range of fields for which the algorithms work and improved running times. What is lost is the rigorous analysis.

We begin with an algorithm of Adleman and DeMarrais ([3]) which we call the IC-NR algorithm. The letters reflect the fact that R is chosen to be a number ring. Let r be the smallest prime congruent to 1 mod n such that the order of p in $(\mathbb{Z}/r\mathbb{Z})^*$ is prime to $(r-1)/n$. Then r has the property that p is inert in the unique subfield of degree n of $\mathbb{Q}(\zeta_r)$, where ζ_r is a primitive rth root of unity. Let $\mathcal{O}$ be the ring of integers of this subfield. Then $\mathbb{F}_q \cong \mathcal{O}/p\mathcal{O}$. Let t_1 be the trace of ζ_r in $\mathcal{O}$ and let $t_2, \ldots, t_n$ be the conjugates of t_1 in $\mathcal{O}$. Then $\{t_j\}$ is a basis for $\mathcal{O}$ over $\mathbb{Z}$ and we can take as representatives for $\mathcal{O}/p\mathcal{O}$ the set

$$T = \{\sum_{j=1}^{n} a_j t_j \,|\, 0 \le a_j \le p-1\}.$$

Now let $R = \mathcal{O}$ and let $\phi : R \to \mathbb{F}_q$ be the quotient map. Let S be the set of prime ideals whose norm is divisible only by primes less than or equal to some given bound. For any $a \in R$, denote by $\overline{a}$ the element in T which is congruent to a mod p. Let τ and β be elements of T such that $\phi(\tau) = t$ and $\phi(\beta) = b$. The rest of the algorithm is the same as the IC-PR algorithm, except that the obstruction group is more serious, as the class group and the unit group of $\mathcal{O}$ both contribute to the obstruction in potentially significant ways.

Adleman and DeMarrais handle this problem by introducing character signatures. Let J be a prime ideal of $\mathcal{O}$ such that $(\mathcal{O}/J)^*$ contains a cyclic subgroup of order $q-1$. Let σ be a generator of this subgroup. Then the character signature of an element $\gamma \in \mathcal{O}$, with respect to the pair (J, σ), is the exponent e determined by the congruence

$$\gamma^{\frac{|(\mathcal{O}/J)^*|}{q-1}} \equiv \sigma^e \bmod J.$$

Notice that the values a character signature takes on are naturally seen to be in $\mathbb{Z}/(q-1)\mathbb{Z}$. Notice also that if γ is a $(q-1)$st power then its character signature is 0. Conversely, one might hope that if an element $\gamma \in V$ has character signature 0 with respect to many ideals J then it is a $(q-1)$st power. Indeed, if one assumes independence of the character signatures, one can make a precise estimate as to how many ideals are necessary to guarantee that γ is a $(q-1)$st power. Assume then that this number is m and that $\rho_1, \ldots, \rho_m$ are distinct character signatures. By including the values $\rho_k(\delta)$ in the vector v_δ and $\rho_k(\gamma)$ in the vector w_γ, where $k = 1, \ldots, m$, we obtain after the linear algebra is performed elements μ and ν which must be $(q-1)$st powers because they are in V and have value 0 for sufficiently many character signatures. Of course, computing a character signature in this case is not a trivial matter since the cardinality of the finite field $\mathcal{O}/J$ is q or bigger. The trick here is to restrict to ideals J of degree one. Then $\mathcal{O}/J$ is a prime field and the algorithm mentioned in the previous section for $n = 1$ can be used.

The result is an algorithm with a conjectured expected running time of $L_q[1/2; 2 + o(1)]$ so long as $p > n$ as $q \to \infty$. The analysis depends on the GRH as well as reasonable, though unproven, assumptions concerning the number of smooth elements in T and properties of character signatures. Nonetheless, when this algorithm is combined with the IC-PR algorithm, we obtain the first example of an algorithm with a conjectured running of $L_q[1/2; c + o(1)]$ for $q \to \infty$ without constraint.

Perhaps the most dramatic development in the last few years has been the appearence of the generalized number field sieve (GNFS) and the function field sieve (FFS), both of which have, under certain assumptions about q, a conjectured expected running time of $L_q[1/3; c + o(1)]$ for $q \to \infty$ and both of which have been implemented for some special fields. The impetus for their discovery was the development of the number field sieve factoring algorithm ([36]). Indeed, soon after its appearence, Gordon ([29]) realized that the number field sieve could be modified to compute discrete logarithms in prime fields. The crucial idea is to let R be a number ring with an assortment of special properties that diminish the size of the elements that are being tested for smoothness.

We describe first how to construct R in the number field case. Given q, let $\mathcal{O}$ be the ring discussed in the IC-NR algorithm with $\mathcal{O}/p\mathcal{O} \cong \mathbb{F}_q$. Assume that $f \in \mathcal{O}[X]$ is a monic, irreducible polynomial of degree d for which $f(b) \equiv 0 \bmod p$ in $\mathcal{O}$. Assume also that the constant term of f is smooth in $\mathcal{O}$, with respect to some bound B, in the sense of the IC-NR algorithm. Let β be a root of f and let $R = \mathcal{O}[\beta]$. Let K be the field of fractions of R and let A be the ring of integers of K. Let P be the prime ideal of R generated by p and $\beta - b$. Then $R/P \cong \mathbb{F}_q$ and we let $\phi : R \to \mathbb{F}_q$ be the quotient map. Let S be the set of prime ideals in A whose norm is divisible only by primes less than or equal to B. In this case, an element of R is smooth if it generates an ideal in A which factors over S. Notice that $\phi(\beta) = b$ and that β is smooth since its norm is the smooth constant term of f. Let $\tau \in R$ be such that $\phi(\tau) = t$ and assume that τ is also smooth. In theory and practice, this assumption is not costly. The algorithm now proceeds by using sieving techniques to find pairs of smooth elements of the form $(c - d\beta, c - db)$ with $c, d \in \mathcal{O}$. For each pair, we construct the associated exponent vectors using the factorizations over S of the ideals these elements generate. Since τ is smooth we also include the pair (τ, τ) where the associated vectors are obtained by considering in one case, the factorization of τ as $\tau \cdot 1$ and in the other, the factorization of (τ) over S.

We now adjust the second stage of the algorithm to account for the fact that, as with the IC-NR algorithm, the units and class group of R can create a substantial obstruction group. If one computes enough character signatures for each smooth pair found and includes these values in the linear algebra, then the obstruction group can be killed. In the case that $n = 1$, however, this approach is not a good one since computing the character signatures is harder than the original problem. Even if this case could be overcome, the use of character signatures for larger n does not seem practical. An alternative, however, is proposed in [58] and [59].

Let l be a prime dividing $q - 1$ and let Γ be the multiplicative subset of A consisting of those elements with norm not divisible by l. Assume that l does not ramify in A. For each prime ideal ℓ lying above l in A, let $\epsilon_\ell = |(A/\ell)^*|$. Then for all $\gamma \in \Gamma$,

$$\gamma^\epsilon \equiv 1 \bmod l,$$

where ϵ is the least common multiple of the ϵ_ℓ. We can, therefore, define maps $\lambda_j : \Gamma \to \mathbb{F}_l$ by means of the congruence

$$\gamma^\epsilon - 1 \equiv \sum_{j=1}^{dn} \lambda_j(\gamma) b_j l \bmod l^2,$$

where $\{b_1, \ldots, b_{dn}\}$ is a basis for A over $\mathbb{Z}$. These maps are logarithmic in the sense that $\lambda_j(\gamma\gamma') = \lambda_j(\gamma) + \lambda_j(\gamma')$. In fact, it is possible to formulate the map $\lambda : \Gamma \to \mathbb{F}_l{}^n$ given by the λ_j as an approximation of the l-adic logarithm.

Returning to the GNFS, we make the following modifications. We only consider $c - d\beta$ and $c - db$ whose norms are prime to l. We include in the exponent vectors the values of the maps λ_j. Though there are only dn of these maps, one can show that this is enough to handle the obstruction group. Finally, we do the linear algebra modulo a suitable power of l so as to kill the l-part of the class group. The result is an algorithm which produces an integer e which does not equal $\log_t b$ but is congruent to this logarithm modulo l. However, since the algorithm can be used for all large l dividing $q - 1$ and in fact can be modified to compute $\log_t v$ modulo prime powers dividing $q - 1$, and since $\log_t v$ modulo a small prime power is easily computed using the Silver-Pohlig-Hellman method, we have a way to compute $\log_t v$ modulo all prime power factors of $q - 1$. The Chinese remainder theorem can now be used to find the logarithm.

The GNFS has a conjectured expected running time of $L_q[1/3; (64/9)^{1/3} + o(1)]$, for $q \to \infty$ so long as for any $\epsilon > 0$, $n < (\log p)^{1/2-\epsilon}$. This is conjecturely the same time needed to factor an integer the size of q using the NFS factoring algorithm. When the constraint on p and n is not met, the elements being tested for smoothness are too large to preserve a running time of $L_q[1/3; c + o(1)]$ for any constant c.

Though the GNFS is a very recent algorithm, a special case was already described by Coppersmith, Odlyzko, and Schroeppel in 1986. In fact, the Gaussian integer method they present can be thought of as the number field sieve in the case that $n = 1$ and R is taken to be a quadratic imaginary number ring.

We turn now to Adleman's function field sieve ([2]). The description of this algorithm is exactly the same as that of the GNFS except that the ring $\mathcal{O}$ is replaced by the ring $\mathbb{F}_p[X]$, the ideal pO is replaced by a prime ideal generated by an irrecucible polynomial of degree n, the set S is the set of ideals in A which divide the ideals in the factor base of the IC-PR algorithm, and the method of handling the obstruction group is different. In this case, there are non-archimedian valuations of K that do not correspond to prime ideals of A but to the primes at infinity. These valuations can be thought of as extensions of the degree map from $\mathbb{F}_p[X] \to \mathbb{Z}$. By putting an extra condition on the polynomial whose root generates R over $\mathbb{F}_p[X]$, Adleman forces A to have only two primes at infinity. The values of these valuations are now included in the exponent vectors and the linear algebra is done modulo a high enough power of $q - 1$ so as to kill the class group, or more precisely the degree 0 part of the Picard group. The result is that one obtains a relation of the sort $b = at^e$, with $a \in \mathbb{F}_p{}^*$. The computation of $\log_t a$ is then an extra step, but one which requires comparatively little time.

The analysis of the running time for the FFS proceeds exactly as that of the GNFS, producing the same conjectural expected value of $L_q[1/3; (64/9)^{1/3} + o(1)]$. Soundararajan's recent work on the probability that a polynomial is smooth can be used to show that this running time heuristically holds so long as $n \geq (\log p)^2$ as $q \to \infty$. When this inequality does not hold, the set S, which must contain at least one element for each linear polynomial in $\mathbb{F}_p[X]$ and consequently has cardinality at least p, is too large. Thus, a gap exists between the ranges of fields for which the FFS and GNFS yield a $L_q[1/3; c + o(1)]$ algorithm and the problem of finding an algorithm with such a conjectured running time for all fields remains open. Nonetheless, the FFS and GNFS can be applied to the fields in the gap. The result is an algorithm, which like the combination of the IC-PR and IC-NR algorithms, has a conjectured expected running time of $L_q[1/2; c + o(1)]$.

Finally, we note that the algorithm of Coppersmith for computing logarithms in $\mathbb{F}_q$ when $q = 2^n$, can be viewed as a special case of the FFS. We assume the reader is familiar with the formulation of this algorithm in [18] and use $P(x)$, $R(x)$, h, and k as Coppersmith does there. Now let $f(x, y) = y^k - R(x)$ and let $\phi : \mathbb{F}_2[x][y]/(f) \to \mathbb{F}_p[x]/(P(x))$ be the ring homomorphism induced by sending $y \mapsto x^h$. The pairs that are checked for smoothness in this case of the FFS are easily seen to be the same polynomials checked in Coppersmith's method. Similiarly, the algorithms of Semaev ([62]) can presumably be formulated as particular cases of the FFS, though we have not done so. These methods are designed to compute logarithms in $\mathbb{F}_{p^n}$ when the order of $p \in (\mathbb{Z}/(2n+1)\mathbb{Z})^*$

equals n or $2n$ or when $p^n - 1$ has a small factor not dividing $p^m - 1$ for all $m < n$. Like Coppersmith's algorithm, they exploit special congruences in $\mathbb{F}_p[X]$.

4 Implementations for finite fields

In this section we describe implementations of some of the heuristic algorithms described in the previous section. We begin with the number field sieve (NFS), which has been implemented to compute logarithms in prime fields, and then consider the function field sieve (FFS) in the form Coppersmith's algorithm for fields of characteristic 2. Finally, we discuss separately the implementation of the linear algebra that occurs in all forms of the index calculus method.

In this cases that we consider a prime field, we represent the field as $\mathbb{Z}/p\mathbb{Z}$ where p is prime and follow the convention of representing elements in the field by their least non-negative residue mod p.

4.1 The NFS: the Algorithm of Coppersmith–Odlyzko–Schroeppel.

In 1991, Odlyzko and LaMacchia ([35]) implemented the Guassian integer method designed by Coppersmith, Odlyzko and Schroeppel (COS). Their implementation was the first to show the weakness of a real cryptographic protocol. The so–called Sun security option is built into the remote procedure call to authenticate users and machines [66]. The system which is used to do this is based on a combination of the Needham–Schroeder protocol [50], which uses DES, and a public key cryptosystem which is a modification of the Diffie–Hellman key exchange protocol [22]. The latter system depends on the difficulty of computing logarithms in $(\mathbb{Z}/p\mathbb{Z})^*$, where p is the 58–digit prime

$$p = 5213619424271520371687014113170182341777563603680354416779.$$

In this case, $(p-1)/2$ is prime. The computations for the system take place within the subgroup of squares mod p generated by the element 3. The problem addressed in [53] was the challenge provided by M. Shannon of Sun Microsystems to compute $\log_3 b$, where $b = 3088993657925229173047110405354521151032325819440498983565$.

As we have seen, the COS–algorithm depends on the choice of a quadratic imaginary number ring R. In the implementation for the Sun challenge, this ring was obtained by adjoining to $\mathbb{Z}$ a root of the polynomial $X^2 + 2$. The factor base consisted of 96321 primes in $\mathbb{Z}[\sqrt{-2}]$. The pairs that were tested for smoothness were of the form $(V(c - d\sqrt{-2}), cV - dT)$, where V and T are chosen so that $T^2 + 2V^2 \equiv 0 \bmod p$. Letting V^{-1} denote the inverse of V mod p, we note that the elements in the pair map to the same element under the quotient map $\phi : \mathbb{Z}[\sqrt{2}] \to \mathbb{Z}[\sqrt{2}]/(p, \sqrt{2} - TV^{-1})$.

In order to compute the logarithm of b, an integer k was found such that

$$b \cdot 3^k \equiv \frac{x}{y} \bmod p$$

where x and y are integers close to $\sqrt{p}$ and divisible by relatively small primes (bounded by 10^{10}). Then the logarithm of each prime factor r of x and y was computed. According to the description in §3, in order to do this r should be smooth. To get around this requirement, a search was done to find c and d such that $r|(cV - dT)$ and both $V(c - d\sqrt{-2})$ and $h = (cV - dT)/r$ are smooth. Using the exponent vector associated to the resulting pair $(V(c - d\sqrt{-2}), rh)$, and solving a slightly different linear system than the one described in §3, one obtains the logarithm of r.

The search for smooth pairs, using sieving techniques, required approximately 1200 hours of computing time. For the linear algebra a variation of Gaussian elimination called structured Gaussian elimination was used to obtain a dense linear system which the Lanczos algorithm then solved in 44 hours ([34]). Both computations were run on a Silicon Graphics 4D-220 having four processors each rated at about 18 mips. In total, less than 2.5 mips-years were spent. According to Odlyzko ([52]), if the computing time were increased by a factor of 800, which would keep it well within the bounds of the amount of computing time used for factoring integers, the implementation should succeed in computing discrete logarithms in a field given by a prime of over 350 bits.

4.2 The NFS: general prime fields

The implementation [68] set a new record when on September 29, 1995 the logarithms of $2, 3, 5, 7, 11, 13, 17, 19, 23$, and 29 to the base 7 were all computed in the field $\mathbb{Z}/p\mathbb{Z}$, where

$$p = 31081938120519680804196101011964261019661412191103091971180537759.$$

In this case p has 65 decimal digits and $l = \frac{p-1}{2}$ is prime.

For these computations, the implementation used a version of the number field sieve in which the monic polynomial defining the number field is replaced by a non-monic polynomial. Before describing it, we consider the monic case in more detail. Let $f(X) \in \mathbb{Z}[X]$ be a monic, irreducible polynomial with a root β. Assume c and d are integers and note that a factorization in $\mathbb{Z}[\beta]$ of the ideal $(c - d\beta)$ is easily determined from the factorization of the norm of $c - d\beta$. In fact, if the norm factors as $\prod p_i^{e_i}$ then $(c - d\beta) = \prod P_i^{e_i}$, where P_i is a degree one prime ideal lying above p_i. If one is careful to avoid those primes in $\mathbb{Z}[\beta]$ which split in the full ring of integers A of $\mathbb{Q}(\beta)$, then the factorization of the norm actually reveals the prime factorization of $(c - d\beta)$ in A. As indicated in §3, this factorization is what is needed to construct an exponent vector. Since the norm of $c - d\beta$ is given by the polynomial

$$N(c - d\beta) = d^n f(c/d),$$

values can be checked for smoothness using a sieve. The trick in the non-monic case is to make use of the the values $d^n f(c/d)$ again, even though β is no longer integral and these values no longer reveal the ideal factorization of $c - d\beta$.

Assume now that $f(X) = \sum_{i=0}^{n} a_i X^i$ is in $\mathbb{Z}[X]$, with $a_n > 1$, and assume that $f(X)$ has a root $m \bmod p$. Let β be a root of $f(X)$ and notice that $\omega = a_n\beta$ is a root of the corresponding monic polynomial $g(x) = x^n + \sum_{i=0}^{n-1}(a_n)^{n-i}x^i$. Let $R_1 = \mathbb{Z}[\omega]$, let $R_2 = \mathbb{Z}[\beta] \cap \mathbb{Z}[\beta^{-1}]$, and let A be the ring of integers of $\mathbb{Q}(\omega)$. In this version of the number field sieve we search for pairs of smooth elements in R_1 of the form $(a_n(c - d\beta), a_n(c - dm))$ and then proceed as described in §3, with the map ϕ being the quotient map from $R_1 \to R_1/(p, \omega - a_n m)$. By smooth, however, we mean something a little different than before. To begin with, we include in the factor base the primes dividing a_n. Thus we need only test $c - d\beta$ and $c - dm$ for smoothness. Furthermore, we consider $c - dm$ as an element of $\mathbb{Z}$. In other words, we include in S, the rational primes less than or equal to some given bound and look for $c - dm$ to factor over this set. Finally, we call $c - d\beta$ smooth if $d^n f(c/d)$ is divisible only by primes less than or equal to a second bound B. As described in [15], the factorization of $d^n f(c/d)$ reveals information about the R_2-module $(R_2 + R_2\beta)/R_2(c - d\beta)$. In particular, it tells us, the number of factors in a composition series of this module that are isomorphic to R_2/P, where P ranges over the prime ideals of R_2 of degree 1 lying over the rational primes dividing $d^n f(c/d)$. For those P for which $P \cdot A$ is prime in A and for which $|R_2/P|$ is prime to a_n, this number is exactly the exponent to which $P \cdot A$ appears in the prime ideal factorization of $(a_n(c - d\beta))$, considered now as an ideal of A. Thus, if one constructs in a straightforward manner an exponent vector for $c - d\beta$ using the exponents appearing in the factorization of $d^n f(c/d)$, and one makes some adjustments for the primes dividing a_n and the few primes of R_2 that factor in A, one can proceed with the linear algebra.

For the computation under consideration, the polynomial defining the number field was

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

There were 16954 prime ideals of A and 3000 rational primes in the factor base. In particular, the factor base included those elements whose logarithm was computed. This fact disposed of the need to find a smooth pre-image under ϕ of each of these elements.

The sieving interval for the search to find smooth pairs $(c - d\beta, c - dm)$ with $m = 13277827521354825$ was

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

The sieving procedure used the idle time of 130 workstations and took 5 y 116 d 15 h 36 m (mips). The computation was distributed among them with the Library for Parallel Systems LiPS [63].

The solution of the 20442×19957 linear system mod l was done on a Paragon machine at the KFA in Jülich/Germany in 38 hours on 50 nodes by using the Lanczos implementation discussed below and in [20]. The solution of x mod 2 was easy to obtain by using the Silver-Pohlig-Hellman algorithm.

4.3 The NFS: special prime fields

The implementation [68] has also been used to compute the logarithms of $3, 5, 11, 23, 31, 67, 7351$, and 11287 to the base 7 in $\mathbb{Z}/p\mathbb{Z}$, where

$$p = \frac{739 \cdot 7^{149} - 736}{3}.$$

This is the field presented by McCurley [43] in his challenge problem.

For a prime the size of p, the polynomial used to generate the number field for the NFS should optimally have a root mod p of size $p^{1/5}$. The polynomial used in this case was $f(X) = 739X^5 - 5152$, which does have such a root, namely 7^{30}. Notice that the special form of p is what allows for the construction of a suitable polynomial with such small coefficients. Let β be a root of $f(X)$, let $R_1 = \mathbb{Z}[\omega]$ where $\omega = 739\beta$, and let ϕ be the quotient map from $R_1 \to R_1/(p, \omega - 739 \cdot 7^{30})$. The algorithm now proceeds as described in the previous subsection. The small size of the coefficients of f are clearly a great advantage, for the smaller the values of $d^n f(c/d)$ are, the more likely that they are divisible by only small primes.

For the computation of the logarithms listed above, a factor base containing 40,000 elements was used. As before, each element whose logarithm was computed was in the factor base. As a result, there was no need to find smooth pre-images under ϕ of these elements. The sieving interval for c and d was

$$\begin{aligned} -15 \cdot 10^6 \leq c &\leq 15 \cdot 10^6 \\ 1 \leq d &\leq 10^6. \end{aligned}$$

After using 110 mips years of idle time on 110 Sparc workstations, a sparse matrix of size 40015×40000 was constructed. The solution of the system modulo the 126–digit prime factor of $p-1$ was done on three Sparc 20 stations in a month using the Lanczos algorithm [20].

There remains the McCurley challenge, namely to compute the logarithm to the base 7 of

$$\begin{aligned} b = {} & 127402180119973946824269244334322849749382042586931621654557 \\ & 735290322291467909599868186097881304659516645545814428058807 6 \\ & 766033781. \end{aligned}$$

Using the congruence

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

where

$$\begin{aligned} s = {} & 3 \cdot 5 \cdot 11 \cdot 23 \cdot 11287 \cdot 10547587 \cdot 2916781859 \cdot 2276186878294984013237 3 \\ & \cdot 51337921071904669 \end{aligned}$$

$$\begin{aligned} t = {} & 31 \cdot 67 \cdot 7351 \cdot 402869 \cdot 2599909498829 \cdot 3598631011739 \\ & \cdot 77731271923481246820848221, \end{aligned}$$

we can reduce the problem to that of computing the logarithms of the relatively small factors of s and t. Indeed, as we have seen, the logarithms of the smallest factors have already been computed. In order to compute the logarithm of any of the bigger prime factors by means of the number field sieve with R_1 given as above, one must find a smooth element in R_1 which is mapped to that prime by ϕ. No good method has yet been found to accomplish this. If one is willing to give up the attractive polynomial $f(X) = 739X^5 - 5152$, then there is a method, described in Gordon's paper ([29]), to find an alternative polynomial which will generate a suitable number ring. However, it is not yet feasible to pursue this route for the primes listed here. The problem is that the coefficients of the polynomials are too big. As a result, the factor base has to be increased in order to find enough smooth pairs in the corresponding number ring. Unfortunately, the size of the factor base then is too great for the available implementations of linear algebra mod $p - 1$. We note that current NFS factoring efforts do, in fact, use number rings given by polynomials with coefficients of the size we are discussing. In this case, however, the linear algebra is done mod 2.

4.4 The FFS: Coppersmith's $\mathbb{F}_{2^n}$–algorithm

Abandoning the general formulation of the function field sieve, we follow Coppersmith's description of his algorithm in [18]. Let the finite field $\mathbb{F}_{2^n}$ be represented as $\mathbb{F}_2[X]/(f(X))$, where $f(X)$ is an irreducible polynomial of the form $X^n + f_1(X)$. For the sake of convenience, the degree of f_1 is chosen as small as possible. Let h and r be constants. The algorithm proceeds by searching for polynomials $u_1(X)$ and $u_2(X)$ of degree $d_1, d_2 \approx n^{1/3}$ respectively, so that

$$w_1 = u_1(X)X^h + u_2(X) \tag{1}$$

is smooth and $w_1(X)^{2^r}$, reduced modulo $f(X)$, is also smooth. By setting r such that $2^r \approx n^{1/3}$, and h to the smallest integer above $n2^{-r}$, the degree of the polynomials being tested for smoothness are of order $n^{2/3}$.

In [28], Gordon and McCurley describe a sieving method they used to find $u_1(X)$ and $u_2(X)$ such that $w_1(X)$ is smooth. Let g be an irreducible polynomial in the factor base. A polynomial of the form (1) which is divisible by g can easily be computed by choosing some polynomial $u_1(X)$ and setting $u_2(X)$ equal to the residue of smallest degree of $u_1(X)X^h$ modulo g. To find more multiples of g of the form (1), one can now add $g(X) \cdot u_3(X)$ to u_2, for every u_3 of degree at most t, where t is appropriately chosen that the generated u_2's are still of degree approximately $n^{1/3}$. The crucial idea here is to enumerate all $g(X) \cdot u_3(X)$ without carrying out the multiplication. The u_3 are represented by bit strings of size t. A Gray code of dimension t is an enumeration G_i, $0 \leq i \leq 2^t - 1$, of all bit strings of size t provided that G_i and G_{i+1} only differ by one bit. There is a recursive and an iterative description of constructing Gray codes of arbitrary dimension. By utilizing the fact that G_i and G_{i+1} differ at bit $l(i)$, where $l(i)$ is the position of the lowest '1'–digit in the binary expansion of i, one can construct a sequence of polynomials $u_{3,i}$ simply by means of the equation

$$u_{3,i+1}(X) = u_{3,i}(X) + g(X) \cdot X^{l(i)} \quad 0 \leq i \leq 2^t - 1.$$

Note that the latter multiplication can be obtained by a left shift of the representation of g. We thus obtain an efficient way to find pairs $u_1(X), u_2(X)$ such that $u_1(X)X^h + u_2(X)$ is divisible by g. Using this method for all g in the factor base produces the smooth polynomials we seek.

Gordon and McCurley have computed logarithms in the the factor base for $n \in \{227, 313, 401\}$. For $n = 401$, it took 111 hours on 1024 processors of an nCUBE–2 to find the relations and 33 hours on 32 processors of an iPSC860 to solve the 117164×58636 linear system. The linear algebra step was accomplished by first reducing the size of the system via structured Gaussian elimination and then applying a parallel version conjugate gradient method.

In the case $n = 503$, the sieving has been done. The resulting 361246×210871 system of linear congruences has not been solved modulo the largest prime factor of $2^{503} - 1$, which has 96 decimal digits.

4.5 Linear algebra

For discrete logarithm algorithms, the linear algebra step has always seemed to be a serious bottleneck. In fact, in order to avoid difficulties in the linear algebra step, the size of the factor based is generally chosen to be smaller than would be optimal for the sieving step. For background on the linear algebra, we refer the reader to [34], [43], [51].

Due to improvements in the sieving step, such as the large prime variations described in [23], the *weight* (the number of non zero elements) of the matrices that occur in the linear algebra stage has greatly increased in recent implementations. As a result, the systems are much more difficult to solve either with structured Gaussian elimination ([34]) or with Krylow subspace algorithms (Lanczos, conjugate gradient, or Wiedemann)([27]). Indeed, the complexity of all these methods depends on both the dimension n and the weight ω of the system. More precisely, it is in $O(n^2 + \omega)$. The sensitivity to weight in the case of structured Gaussian elimination is reflected in the fact that, in [34], a reduction in dimension of approximately 90% on systems with an average number of

15.5 entries per equation was achieved, whereas in the systems which arise from NFS computations and which have an average number of approximately 390 entries per equation, a reduction in the dimension of only 50% can be achieved. Ordinary Guassian elimination, in contrast, only depends on the dimension of the system. Nonetheless, the practicability of a combination of structured Gauss and ordinary Gaussian elimination suffers from enourmous space requirements.

Currently, our approach is to combine structured Gaussian elimination with one of the Krylow subspace algorithms. By employing Gaussian elimination first, we can decrease the dimension of the linear system, but due to the performed operations the weight of the system increases. While the expected running time of the Krylow subspace algorithms decreases we keep on doing structured Gaussian elimination and then we solve the compactified system with a Krylow subspace algorithm. We prefer the Lanczos algorithm to solve the compactified system because it is quite fast and has moderate space requirements. We give a description of this method following [53].

Suppose we want to solve the sytem $Ax = w$ for a column n-vector x, where A is a symmetric $n \times n$ matrix, and w is a given column n-vector. Let

$$\begin{aligned} w_0 &= w, \quad v_1 = Aw_0 \\ w_1 &= v_1 - \frac{\langle v_1, v_1 \rangle}{\langle w_0, v_1 \rangle} w_o \end{aligned}$$

and then for $i \geq 1$, define

$$\begin{aligned} v_{i+1} &= Aw_i \\ w_{i+1} &= v_{i+1} - \frac{\langle v_{i+1}, v_{i+1} \rangle}{\langle w_i, v_{i+1} \rangle} w_i - \frac{\langle v_{i+1}, v_i \rangle}{\langle w_{i-1}, v_i \rangle} w_{i-1} \end{aligned}$$

The algorithm stops when it finds a w_j such that $\langle w_j, Aw_j \rangle = 0$. This happens for some $j \leq n$. If $w_j = 0$ then

$$x = \sum_{i=0}^{j-1} \frac{\langle w_i, w \rangle}{\langle w_i, v_{i+1} \rangle} w_i$$

is a solution of the system. For more details, for example on how to deal with non-symmetric matrices or self-conjugacy we refer to [53]. In every iteration $i > 1$ we have to compute the vector $v_{i+1} = Aw_i$ and 3 inner products. We were able to reduce the number of inner products from 3 to 2 inner products per iteration, by using the identity $\langle v_{i+1}, v_i \rangle = \langle w_i, v_{i+1} \rangle$ in the ith Lanczos iteration. For details see [20]). This reduces not only the running time per iteration , it also reduces the space requirements. The vector v_i is no longer needed in the i-th iteration.

There exist other efforts to reduce the weight of the systems that have to be solved. In [21] a method is described to reduce the weight of relations that are composed with the large prime variation. Slightly more sieving is also useful to reduce the weight of the systems. By doing more sieving, you get relations which are obtained by combining fewer *partial relations* (relations with up to 4 large primes) and therefore the weight of the matrix is decreased. As an example we list some data collected during the precomputation step of the McCurley challenge [43]. We list the number of partial relations that are needed to build the matrix. We counted this number at three different points during the sieving step. Even at the first point we had twice as many relations as were needed to build the matrix. We also list the maximum number of partial relations that were needed to compose an equation.

point	# partial relations	maximum
I	1 664 276	78
II	1 113 781	48
III	567 592	23

In order to get a rough estimate on how long it would take to solve systems that arise from a discrete logarithm problem modulo an arbitrary 129 digit prime, we did some calculations based on our experience. We assume that in all cases the average weight per equation is ≈ 400 and that 15 solutions are needed. In the first column the dimension of the system is listed and the second column holds the expected running time on 136 nodes of the Paragon machine in Jülich.

dimension	running time
125 000	21 days
250 000	81 days
500 000	335 days
1 000 000	1250 days

5 Class groups

We discuss some recent results concerning the computation of discrete logarithms in the class group of an imaginary quadratic order. We do not consider class groups in other settings. For recent work on computing discrete logarithms in the class group of a function field we refer the reader to [48]. For a discussion of a Diffie-Hellman type protocol which depends for its security in part on the difficulty of computing discrete logarithms in class groups of general number rings we refer the reader to [13].

Let Cl be the class group of an imaginary quadratic order of discriminant D. We begin by reporting that a deterministic version of Shank's baby step/giant step algorithm has been implemented which is able to compute logarithms in the case that $D = -4(10^{20}+1)$ in about 100 seconds on a Sparc Station 20 [11]. The modification of Shank's method used for this computation allows one to prescribe the number of baby steps. This has a significant advantage in class groups, because in this case the order of a group element is not known in general. Instead of taking the group order as an upper bound, Jacobson and Teske experiment with smaller values and obtain running times, which are considerably better.

We turn now to a subexponential algorithm for discrete logarithms in CL which makes use of the algorithm of Hafner and McCurley [30] for computing the structure of CL. This algorithm uses the index calculus method to find integers $m_1, \dots, m_k$, where $m_i | m_{i+1}$ for $1 \le i \le k-1$, such that

$$Cl \cong \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z},$$

The algorithm depends on the correspondence between ideals of quadratic orders and quadratic forms which is described in Theorem 5.24 of [17]. Indeed, all computations in the class group are done with forms. To begin with a set S of "small" prime forms is found. The set S is taken sufficiently large so that, if the Generalized Riemann Hypothesis (GRH) is true, the elements of S generate CL (see [5]). The algorithm proceeds to look for powers of prime forms in S whose corresponding reduced form factors over S. When this occurs one obtains a relation in the class group. The factorization over S of the forms in this relation gives an exponent vector. Taken together, these vectors give a matrix A whose determinant is a multiple of the class number. Let h^* be a lower bound for the class number obtained by computing finitely many terms of the analytic class number formula ([61]). When $\det A < 2h^*$, one has the correct class number. In this case, the Smith–Normal–Form of A reveals the group structure of Cl. In fact, one has enough information to obtain generators $\gamma_1, \dots, \gamma_k$ for the cyclic components of CL, as products of powers of the original prime forms of S. The algorithm now to compute $\log_t b$ proceeeds by expressing t and b as products over S and using the information produced by the Hafner-McCurley algorithm to transform these products into products over the generators γ_i. At this point the original discrete logarithm problem is easily reduced to computing linear congruences mod $|\langle\gamma_i\rangle|$ for all i. We refer the reader to [12] for the details as well as the analysis that shows that once the algorithm of Hafner and McCurley is complete, one can compute a logarithm in expected time $L_D[1/2; 1/2 + o(1)]$ for $D \to \infty$. The algorithm of Hafner and McCurley is also subexponential, with an expected running time of $L_D[1/2; \sqrt{2} + o(1)]$.

The algorithm just described had been implemented. The largest discriminant for which logarithms were computed was $4 \cdot 2^{2^7}$, which has 40 decimal digits. The computation of the class group took 511978 seconds. The computation of a particular discrete logarithm took only 114 seconds on a Sun 4/60 Sparc Station 1.

6 Open questions

The index calculus has had remarkable success in computing discrete logarithms in finite fields. It has also been modified to work in some other settings. In this section we present some unsolved problems concerning both the index calculus method and the discrete logarithm problem more generally. Many of the questions we give are modifications of those on McCurley's list of open questions in [43]. For additional problems, about which we had nothing new to add, we refer the reader to this list.

- Can the equivalence of the discrete logarithm problem and the Diffie-Hellman problem be shown for those primes not covered by the the work described in §2.1? In particular, can the result of §2.1 be extended?

- Can one find an algorithm which has a running time of $L_q[1/3; c + o(1)]$, where c is a constant and $q \to \infty$ with no conditions on p and n? In particular, can one fill the gap that presently exists between the number field sieve and the function field sieve? At present the gap, roughly speaking, consists of those pairs p, n such that $(logp)^{1/2} < n < (logp)^2$. For more details, see [59].

- Can one prove the running times conjectured for the heuristic algorithms described in this paper?

- Is there a practical version of the number field sieve which can take advantage of primes of the form $r^e - s$, where r and e are small? We refer the reader to [36] for a discussion of the effectiveness of the number field sieve for factoring integers of this form. More specifically, can the number ring described in §4.3 be used to solve the McCurley challenge?

- Can one improve the current methods for solving a system of linear equations modulo a large prime. In particular, is there an algorithm for which the work can be cheaply distributed over a cluster of workstations?

- Can one find a subexponential discrete logarithm algorithm for $\mathbb{F}_p$ whose running time does not depend on p but on the largest prime factor of $p - 1$. Such an algorithm might be useful for breaking systems such as the DSS scheme which uses a subgroup of size 160 digits in a field of size 512 digits.

- Is there a subexponential algorithm for computing discrete logarithms in the group of points of an elliptic curve over a finite field? In [46], Menezes, Okamoto, and Vanstone show how to use the Weil pairing to reduce the discrete logarithm problem on an elliptic curve over a finite field to the same problem in an extension of the finite field. In the cases that the degree of this extension is sufficiently small, their reduction yields a subexponential algorithm for computing discrete logarithms on the curve. Recently, Balasubramanian and Koblitz ([6]) have shown that it is very unlikely this approach will yield a subexponential algorithm for a randomly chosen curve suitable for cryptography.

- Are there other groups which, like the group of points on an elliptic curve, are easy to compute in and for which the discrete logarithm problem is apparently difficult? In this regard, we note that there is a subexponential algorithm for computing logarithms in the Jacobian of a hyperelliptic curve over a finite field, provided its genus is large enough ([4]).

7 Acknowledgements

The authors wish to thank Andrew Odlyzko, Johannes Buchmann, Dan Gordon, Kevin McCurly, Alfred Menezes, Stefan Wolf, and especially Carl Pomerance for their many helpful suggestions.

References

1. L. M. Adleman, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, Proc. 20th IEEE Found. Comp. Sci. Symp., pp. 55-60, 1979
2. L. M. Adleman, *The function field sieve*, Algorithmic number theory (L.M. Adleman and Ming-Deh Huang, eds.), Lecture Notes in Computer Science 877, Springer-Verlag, pp 108-121, 1994
3. L. M. Adleman, J. DeMarrais, *A subexponential algorithm for discrete logarithms over all finite fields*, Math. Comp. 61, pp 1-155, 1993
4. L.M. Adleman, J. DeMarrais, M.-D. Huang, *A subexponential algorithm for discrete logarithms of large genus hyperelliptic curves over finite fields*, Lecture Notes in Computer Science 877, Springer-Verlag, pp 28-40, 1994
5. E. Bach, *Explicit bounds for primality testing and related problemes*, Math. Comp. 55, pp. 355-380, 1990
6. R. Balasubramanian, N. Koblitz, *The improbability that an elliptic curve has subexponential discrete logarithm problem under the Menezes–Okamoto–Vanstone Algorithm*, in preparation
7. I. Biehl, J. Buchmann, Th. Papanikolaou *LiDIA – A library for computational number theory*, Universität des Saarlandes, preprint, 1995
8. B. den Boer, *Diffie–Hellman is as strong as discrete log for certain primes*, Advances in Cryptology – Crypto '88, Lecture Notes in Computer Science, 403, pp. 530–539, Springer, 1989
9. D. Boneh, R. Lipton, *Algorithms for black box fields and their application to cryptography*, submitted to Crypto '96, preprint, 1995
10. R. P. Brent, *An improved Monte Carlo factorization algorithm*, Nordisk Tidskrift för Informationsbehandling (BIT) 20, pp. 176–184, 1980
11. J. Buchmann, M. Jacobson, E. Teske, *On some computational problems in finite abelian groups*, to appear
12. J. Buchmann, St. Düllmann, *On the computation of discrete logarithms in class groups*, Advances in Cryptology Crypto '90, Lecture Notes in Computer Science 537, 1991
13. J. Buchmann, *Number theoretic algorithms and cryptology*, Proceedings FCT '91, Lecture Notes in Computer Science 529, 1991
14. 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
15. J. P. Buhler, H. W. Lenstra, Jr., C. Pomerance, *Factoring integers with the number field sieve*, The development of the number field sieve (A.K. Lenstra, H.W. Lenstra, Jr., eds.), Lecture Notes in Mathematics 1554, Springer, 1993
16. D. Chaum, E. van Heijst, B. Pfitzmann, *Cryptographically strong undeniable signatures, unconditionally secure for the signer*, Advances in Cryptology Crypto '91, Lecture Notes in Computer Science, 435, pp. 212–216, Springer, 1992
17. H. Cohen, *A course in computational algebraic number theory*, Springer, 1993
18. D. Coppersmith, *Fast evaluation of discrete logarithms in fields of characteristic two*, IEEE Trans. Information Theory IT-30, pp 587-594, 1984
19. D. Coppersmith, A. Odlyzko, R. Schroeppel, *Discrete logarithms in GF(p)*, Algorithmica 1, pp. 1–15, 1986
20. Th. Denny, *A Lanczos implementation for GF(p)*, Universität des Saarlandes, to appear

21. Th. Denny, V. Müller, *On the reduction of composed relations from the number field sieve* , ANTS II, 1996
22. W. Diffie, M. Hellman, *New directions in cryptography.* IEEE Trans. Inform. Theory 22, pp. 472-492, 1976
23. B. Dodson, A. K. Lenstra, *NFS with four large primes: an explosive experiment*, Advances in Cryptology, CRYPTO '95, Lecture Notes in Computer Science 963, Springer Verlag, pp. 372 - 385, 1995
24. St. Düllmann, *Ein Algorithmus zur Bestimmung der Klassengruppe positiv definiter quadratischer Formen*, PhD thesis, Saarbrücken, 1991
25. T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Information Theory, 31, pp. 469-472, 1985
26. T. ElGamal, *A subexponential-time algorithm for computing discrete logarithms over* $GF(p^2)$, IEEE Trans. Information Theory, 31, pp. 473-481, 1985
27. G. H. Golub, C. F. Van Loan, *Matrix computations*, The Johns Hopkins University Press, 1993
28. D. Gordon, K. McCurley, *Massively parallel computation of discrete logarithms*, Advances in Cryptology – Crypto 92, Lecture Notes in Computer Science 740, Springer, pp 312-323, 1993
29. D. Gordon, *Discrete logarithms in GF(p) using the number field sieve*, SIAM J. Discrete Math., Vol 6, pp. 124–138., 1993
30. J. Hafner, K. McCurley, *A rigorous subexponential algorithm for computation of class groups*, Journal AMS
31. N. Koblitz, *Elliptic curve cryptosystems*, Math. Comp. 48, pp 203-209, 1987
32. M. Kraitchik, *Théorie des nombres*, Gauthier–Villars, Vol. 1, 1922
33. M. Kraitchik, *Recherche sur la théorie des nombres*, Gauthier–Villars, Vol. 1, 1922, Gauthier–Villars, 1924
34. M. LaMacchia, A. Odlyzko, *Solving large sparse linear systems over finite fields*, Advances in Cryptology Crypto '90, Lecture Notes in Computer Science 537, pp. 109-133, Springer, 1991
35. M. LaMacchia, A. Odlyzko, *Computation of discrete logarithms in prime fields*, Designs, Codes and Cryptography 1, pp. 46-62, 1991
36. A. K. Lenstra, H. W. Lenstra, Jr., *The development of the number field sieve*, Lecture Notes in Mathematics 1554, Springer, 1993
37. A. K. Lenstra, H. W. Lenstra, Jr., *Algorithms in number theory*, Technical Report 87-008, University of Chicage, 1987
38. H. W. Lenstra, Jr., C. Pomerance *A rigorous time bound for factoring integers*, J. Amer. Math Soc. 5, pp 483-516, 1992
39. R. Lovorn, *Rigorous, subexponential algorithms for discrete logarithms over finite fields*, Ph.D. Thesis, University of Georgia, June 1992.
40. R. Lovorn Bender, *Rigorous, subexponential algorithms for discrete logarithms in* $GF(p^2)$, SIAM J. Discrete Math., to appear
41. R. Lovorn Bender, C. Pomerance *Rigorous discrete logarithm computations in finite fields via smooth polynomials*, preprint, 1995
42. U. Maurer, *Towards the equivalence of breaking the Diffie–Hellman protocol and computing discrete logarithms*, Advances in Cryptology Crypto '94, Lecture Notes in Computer Science, Springer, 1994
43. K. McCurley, *The discrete logarithm problem*, Cryptology and computational number theory, Proc. Symp. in Applied Mathematics, American Mathematical Society, 1990
44. U. Maurer, St. Wolf, *Diffie–Hellman–oracles*, submitted to Crypto '96, preprint, 1995

45. A. Menezes, *Elliptic curve public key cryptosystems*, Kluwer, 1993
46. A. Menezes, T. Okamoto, S. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing, pp 80-89, 1991
47. V. Miller, *Use of elliptic curves in cryptography*, Advances in Cryptology: Proceedings of Crypto '85 (H.C. Williams, ed.), Lecture Notes in Computer Science 218, Springer-Verlag, pp 417-426, 1986
48. V. Müller, A. Stein, Ch. Thiel, *Computing discrete dlLogarithms in real quadratic congruence function fields of large genus*, 1996
49. National Bureau of Standards, *Digital signature standard*, FIPS Publication 186, 1994
50. R. Needham, M. Schroeder, *Using encryption for authentication in large networks of computers*, Comm. ACM 21, pp. 993-999, 1978
51. A. Odlyzko, *Discrete logarithms in finite fields and their cryptographic significance*, Advances in Cryptology, Proc. Eurocrypt 84 (T. Beth, N. Cot, and I. Ingemarrsson, eds.), Lecture Notes in Computer Science 209, Springer-Verlag, Berlin, pp. 224-314, 1985
52. A. Odlyzko, *Discrete logarithms and smooth polynomials*, Finite fields: theory, applications, and algorithms (Las Vegas, NV, 1993), Contemp. Math 168, Amer. Math. Soc., Providence, R.I., pp 269-278, 1994
53. A. Odlyzko, M. LaMacchia, *Discrete logarithms in GF(p)*, 1991
54. 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
55. J. M. Pollard, *Monte Carlo methods for index computation (mod p)*, Math. Comp. 32, 918–924, 1978
56. C. Pomerance, *Fast rigorous factorization and discrete logarithms algorithms*, Discrete algorithms and complexity (D.S. Johnson, T. Nishizeki, A. Nozaki and H. Wilf, eds.), Academic Press, Orlando, Florida, pp. 119-143, 1987
57. R. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communication of the ACM 21, pp. 120–126, 1978
58. O. Schirokauer, *Discrete logarithms and local units*, Phil. Trans. R. Soc. Lond. A 345, 409–423, 1993
59. O. Schirokauer, *Using number fields to compute logarithms in finite fields*, to appear
60. C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, 4, pp. 161–174, 1991
61. R. Schoof, *Quadratic fields and factorization*, Computational Methods in Number Theory, Mathematisch Centrum Trakt 154, Amsterdam, pp. 235-286, 1982
62. I. A. Semaev, *An algorithm for evaluation of discrete logarithms in some nonprime finite fields*, Math. Comp., to appear
63. Th. Setz, R. Roth, *LiPS: a system for distributed processing on workstations*, SFB 124 TP D5, Universität des Saarlandes, 1992
64. K. Soundararajan, *Asymptotic formulae for the counting function of smooth polynomials*, J. London Math. Soc., to appear
65. D. R. Stinson, *Cryptography in theory and practice*, CRC Press, 1995
66. B. Taylor, D. Goldberg, *Secure networking in the Sun environment*, Proc. USENIX Assoc. Summer Conference, Atlanta, pp. 28-37, 1986
67. D. Weber, *An implementation of the number field sieve to compute discrete logarithms mod p*, Advances in Cryptology – Eurocrypt'95, pp. 95–105, 1995
68. D. Weber, *Computing discrete logarithms with the number field sieve*, ANTS II, 1996

69. J. Zayer, *Faktorisieren mit dem Number Field Sieve*, PhD thesis, Saarbrücken, 1995