THE INDEX CALCULUS METHOD USING NON-SMOOTH POLYNOMIALS

THEODOULOS GAREFALAKIS AND DANIEL PANARIO

ABSTRACT. We study a generalized version of the index calculus method for the discrete logarithm problem in $\mathbb{F}_q$, when $q = p^n$, p is a small prime and $n \to \infty$. The database consists of the logarithms of all irreducible polynomials of degree between given bounds; the original version of the algorithm uses lower bound equal to one. We show theoretically that the algorithm has the same asymptotic running time as the original version. The analysis shows that the best upper limit for the interval coincides with the one for the original version. The lower limit for the interval remains a free variable of the process. We provide experimental results that indicate practical values for that bound. We also give heuristic arguments for the running time of the Waterloo variant and of the Coppersmith method with our generalized database.

1. INTRODUCTION

Let G denote a group written multiplicatively and $\langle g \rangle$ the cyclic subgroup generated by $g \in G$. Given g and $y \in \langle g \rangle$, the *discrete logarithm problem* for G is to find the smallest integer x such that $y = g^x$. The integer x is called the *discrete logarithm* of y in the base g, and is written $x = \log_g y$.

The main reason for the intense current interest on the discrete logarithm problem (apart from being interesting on a purely mathematical level) is that the security of many public-key cryptosystems depends on the assumption that finding discrete logarithms is hard, at least for certain groups. For instance, the security of cryptographic applications such as the Diffie-Hellman key exchange scheme [5], El Gamal's cryptosystem [6], and pseudorandom bit generators [2, 8] depend on the current ability (or inability) to solve the discrete logarithm problem efficiently.

Several groups have been proposed for the implementation of cryptographic systems including the multiplicative group of finite fields, the group of points of elliptic curves over finite fields [14], class groups of number fields [3], and function fields [15], for instance. We focus on the discrete logarithm problem in the multiplicative group of finite fields $\mathbb{F}_q$, where $q = p^n$, p is a small prime and n is large. Lovorn Bender and Pomerance [13] present results when p and n both tend to infinity. Using the isomorphism $\mathbb{F}_q \cong \mathbb{F}_p[x]/(f)$, where f is a monic irreducible polynomial over $\mathbb{F}_p$ of degree n, elements in $\mathbb{F}_q^*$ can be represented as polynomials over $\mathbb{F}_p$ of degree at most $n-1$. The breakthrough in the computation of discrete logarithms in such groups was the development of the *index calculus method*. The basic method has

Date: May 16, 2000.

1991 *Mathematics Subject Classification.* Primary 11Y16, 12E05; Secondary 11T71, 68P25, 68Q25, 94A60.

Key words and phrases. Finite fields, discrete logarithm problem, cryptography, smooth polynomials.

been proven [16] to run in subexponential time of the form

$$\exp\left((\sqrt{2\log p}+o(1))\sqrt{n\log n}\right).$$

We revise the index calculus method in Section 2. As it is shown there and is well-known, the algorithm depends on finding polynomials with all of their irreducible factors with degree not greater than certain bound m, the so-called *smooth* polynomials. In this case, the database of logarithms is formed by all irreducible polynomials of degree smaller than or equal to m. In this paper, we refer to this set of irreducible polynomials as the *standard factor base* or the *smooth factor base*. Until now, the standard factor base was considered to be the natural choice; see, for instance the surveys by Odlyzko [16, 17].

The goal of this work is to formalize this belief that the standard factor base is the best possible (see [16], p. 237). In order to do that, we are interested in developing a generalized version of the index calculus method for the discrete logarithm problem in $\mathbb{F}_q$, when $q = p^n$, p is a small prime and n large, but considering a database formed by the logarithms of all irreducible polynomials of degree between given bounds. We call this the *generalized factor base*. We remark that the standard factor base is a particular case of the generalized factor base. This led us to the study of the number of polynomials over $\mathbb{F}_q$ with all their irreducible factors with degree in an interval. Our companion paper [9] provides the needed estimates; Section 3.2 summarizes the crucial results for our purposes here.

In Section 3, we analyze the basic index calculus algorithm when a generalized factor base is considered. The results are two-fold. On the one hand, we obtain theoretical estimations that are as good as the ones for the standard base (Theorem 3). As in the smooth case, there is not much freedom for the upper limit of the interval. However, the lower limit remains a free variable which can be chosen (almost) at will.

On the other hand, we perform some experiments in Section 6 that suggest that the best choice for the lower limit is one, which corresponds to the standard factor base.

The basic version of the index calculus algorithm has been rigorously analyzed [16]. In order to improve the running time of the method, several variants have been proposed [1, 4]. In Sections 4 and 5, we show that the heuristic arguments for the running time of the Waterloo algorithm and for the Coppersmith variant can be applied for our generalized version.

Finally, we comment that our results also hold when the factor base consists of all monic irreducible polynomials with degree in a set T provided that T contains at least all integers between $c\sqrt{(n\log n)/(2\log p)}$ and $\sqrt{(n\log n)/(2\log p)}$, for $c<1$ a constant.

2. The Algorithm

We briefly describe the index calculus method for computing discrete logarithms in $\mathbb{F}_q$, $q = p^n$, p is a prime, and $n > 1$. The elements in $\mathbb{F}_q$ are represented as polynomials over $\mathbb{F}_p$ of degree smaller than n. Arithmetic is performed modulo a monic irreducible polynomial $f \in \mathbb{F}_p[x]$ of degree n. First we choose a set S of irreducible polynomials over $\mathbb{F}_p$. We call S the *factor base*. We are given p, n, f, a generator g of the multiplicative group of $\mathbb{F}_q$, and a polynomial $h^* \in \mathbb{F}_p[x]$ whose discrete logarithm we want to compute. The method consists of two stages.

Stage 1

1. Choose an integer s in $[1, q-1]$ uniformly at random, and form the polynomial $h \equiv g^s \pmod f$, $\deg h < n$.
2. Check if h factors completely into irreducibles over S. If not, discard it. If it does, say
$$h = \prod_{v \in S} v^{e_v(h)},$$
record the congruence
$$s \equiv \sum_{v \in S} e_v(h) \log_g v \pmod{q-1}.$$

Repeat the above steps until "slightly more" than $\#S$ congruences are obtained. Then solve the system to determine $\log_g v$ for all $v \in S$.

Stage 2

1. Choose an integer s in $[1, q-1]$ uniformly at random; form the polynomial $h \equiv h^* g^s \pmod f$, $\deg h < n$.
2. Check if h factors completely into irreducibles over S. If not, discard it. If it does, say
$$h = \prod_{v \in S} v^{e_v(h)},$$
compute the required discrete logarithm as
$$\log_g h^* \equiv -s + \sum_{v \in S} e_v(h) \log_g v \pmod{q-1}.$$

The method works for any factor base S. The choice of the factor base however, clearly affects the time complexity of the algorithm. For example, a "very large" factor base, that would speed up the second stage, would make the first stage (and thus the whole method) totally inefficient. Until now, it was generally assumed that the optimal choice of a factor base was the set of all monic irreducible polynomials of degrees smaller than or equal to m for a certain parameter m, the smooth factor base, although no proof has been given of this fact.

We consider a more general version, when the factor base consists of all monic irreducibles of degree between m_2 and m_1. Our analysis shows that the important parameter here is m_1. In the next section, we show that the asymptotic running time of the algorithm remains the same as the basic version even if m_2 is of the same order as m_1. This means that the size of the factor base could be smaller, an interesting fact for practical purposes when space is a constraint.

3. Analysis of the index calculus method with generalized factor base

The analysis of the algorithm in Section 2 requires the study of the size of the factor base and the probability of successfully factoring polynomials into irreducibles over S.

3.1. **The Size of the Factor Base.** We start by estimating the size t of the factor base.

Proposition 1. *Let S be a factor base formed by irreducible polynomials over $\mathbb{F}_p$ with degree between m_2 and m_1, $m_2 < m_1$. Then, as $m_1 \to \infty$, the size t of the factor base S is bounded above by*

$$e^{(1+o(1))m_1 \log p}. \tag{1}$$

Proof. It is well-known (see for instance [11], Theorem 3.25) that the number I_k of monic irreducible polynomials of degree k over $\mathbb{F}_p$ is given by

$$I_k = \frac{1}{k}\sum_{d|k} \mu(d) p^{k/d} = \frac{p^k}{k} + \sum_{d|k,\ d>1} \mu(d) p^{k/d}.$$

It is easy to derive the following upper bound

$$\left| I_k - \frac{p^k}{k} \right| = \left| \sum_{d|k,\ d>1} \mu(d) p^{k/d} \right| \leq \sum_{j=1}^{\lceil k/2 \rceil} p^j = \frac{p^{\lceil k/2 \rceil + 1} - 1}{p-1} < 2p^{k/2}.$$

An upper bound on the size of the factor base can be computed as $m_1 \to \infty$

$$\begin{aligned} t &= \sum_{k=m_2+1}^{m_1} I_k < \sum_{k=m_2+1}^{m_1} \left(\frac{p^k}{k} + 2p^{k/2} \right) \\ &\leq \frac{1}{m_2} \sum_{k=m_2+1}^{m_1} p^k + 2 \sum_{k=m_2+1}^{m_1} p^{k/2} = \frac{1}{m_2} \left(\frac{p^{m_1+1}}{p-1} - \frac{p^{m_2+1}}{p-1} \right) + O\left(p^{m_1/2} \right) \\ &\leq \frac{p^{m_1+1}}{m_2} + O\left(p^{m_1/2} \right) = e^{(1+o(1))m_1 \log p}. \end{aligned}$$

□

The quantity needed for the analysis of the algorithm is the size of the system of congruences created in the first stage. It is shown in [12] that if $4t \log p^n$ linear congruences are computed, the probability that the system has full rank is at least $1/2$.

3.2. **The Probability of Success.** We now give an estimate on the number of repetitions needed in both stages until the polynomial h completely factors over the factor base S.

As it will be clear below, we need an estimate for the probability that a random monic polynomial of degree at most $n-1$ factors over S, i.e., it factors into irreducibles of degree greater than m_2, and less than or equal to m_1. The next two theorems provide that estimate (see [9], Theorems 2 and 4). Their proofs are rather technical and in the spirit of analytic number theory. The basic components of the proofs are generating functions for counting the polynomials in question, and the saddle point method for deriving asymptotic results.

The first theorem deals with the case that the lower bound m_2 is a fixed constant (independent of n). We note that the case $m_2 = 0$ corresponds to the smooth factor base used in the original calculus method.

Theorem 1. *The number $N_q(n, m_1, m_2)$ of monic polynomials of degree n over $\mathbb{F}_q$ with all irreducible factors with degree between m_2 and m_1, $m_2 < m_1$, with m_2 fixed and $\log n \ll m_1 \ll n$, satisfies*

$$N_q(n, m_1, m_2) = q^n e^{-(1+o(1))u_1 \log u_1},$$

where $u_1 = n/m_1$.

The second theorem extends the result to the case that both m_1 and m_2 tend to infinity as functions of n.

Theorem 2. *The number $N_q(n, m_1, m_2)$ of monic polynomials over $\mathbb{F}_q$ with all irreducible factors between m_1 and m_2, with $m_1, m_2 \to \infty$, $m_1 e^{-n/m_1} \ll m_2 \leq cm_1$ for any constant $c < 1$, and $2(\log n)^2 \leq m_1 \ll n$ satisfies*

$$N_q(n, m_1, m_2) = q^n e^{-(1+o(1))u_1 \log u_1},$$

where $u_1 = n/m_1$.

The next proposition gives an upper bound on the number of repetitions needed in both stages until the polynomial h completely factors over the factor base S.

Proposition 2. *Let S be a factor base formed by irreducible polynomials over $\mathbb{F}_p$ with degree between m_2 and m_1, with m_2 and m_1 as in the hypothesis of either Theorem 1 or Theorem 2. Then, the expected number of repetitions of Step (2) in both Stage 1 and Stage 2 of the algorithm in Section 2 is bounded above by*

$$e^{(1+o(1))\frac{n}{m_1}\log\frac{n}{m_1}}. \tag{2}$$

Proof. We recall that the polynomial h in the algorithm of Section 2 is a random monic polynomial in $\mathbb{F}_p[x]$ of degree at most $n-1$, and the experiment is a sequence of Bernoulli trials. The expected number of repetitions until we have the first success is one over the probability of success. Thus, a lower bound on the probability of success would provide an upper bound on the expected number of repetitions. Then, we have

$$\begin{aligned} Pr(h \text{ factors over } S) &= \sum_{k=1}^{n-1} Pr(h \text{ factors over } S | \deg h = k) Pr(\deg h = k) \\ &\geq Pr(h \text{ factors over } S | \deg h = n-1) Pr(\deg h = n-1). \end{aligned}$$

It is easy to see that the probability of a random monic polynomial of degree at most $n-1$ having degree exactly equal to $n-1$ is very close to 1. More precisely, we have

$$Pr(\deg h = n-1) = \frac{\#\text{ polynomials of degree } n-1}{\#\text{ polynomials of degree } \leq n-1} = \frac{p^{n-1}}{\sum_{i=1}^{n-1} p^i} > 1 - \frac{1}{p}.$$

It remains to combine this with the probability that a random monic polynomial of degree $n-1$ factors over S given in Theorems 1 and 2. Choosing the parameters m_1 and m_2 so that they satisfy the conditions of these theorems, we can estimate the probability of interest as

$$Pr(h \text{ factors over } S | \deg h = n-1) = e^{-(1+o(1))u_1 \log u_1},$$

where $u_1 = n/m_1$. In fact, one should let $u_1 = (n-1)/m_1$, but the $o(1)$ in the exponent "neutralizes" such small changes. Therefore, the expected number of repetitions is bounded above by

$$e^{(1+o(1))\frac{n}{m_1}\log\frac{n}{m_1}},$$

where the multiplicative term $(1-1/p)^{-1}$ due to the probability that h is of degree $n-1$ is absorbed by the $o(1)$ in the exponent. □

3.3. **Upper Bound on the Running Time.** We have all the quantities needed to analyze the algorithm. In this section we will compute a precise value for m_1, in terms of n, and give an upper bound for the running time of the index calculus method in terms of the function

$$L(n) = e^{(1+o(1))\sqrt{n \log n}}.$$

The main result (Theorem 3) shows that asymptotically our factor base is as good as the standard base used in the basic index calculus algorithm.

As it is the case when the standard factor base is used, the running time of the algorithm is dominated by the first stage. This is when a tradeoff takes place regarding the size of the factor base: large $\#S$ means small number of repetitions (until a useful congruence is found), but many such congruences are needed for the system to be solvable. The complexity of the algorithm is proven in the following theorem.

Theorem 3. *The running time of the algorithm in Section 2 is, as* $n \to \infty$,

$$L(n)^{\sqrt{2 \log p}}.$$

Proof. From Equations (1) and (2) and the fact that $4tn \log p$ congruences have to be generated, where t is the size of the factor base, we conclude that the time to create the system is

$$4n \log p e^{(1+o(1))\left(m_1 \log p + \frac{n}{m_1} \log \frac{n}{m_1}\right)} \sim e^{(1+o(1))\left(m_1 \log p + \frac{n}{m_1} \log \frac{n}{m_1}\right)}.$$

Furthermore, if a method for sparse linear systems is used like the method proposed by Wiedemann [18] (see also [16]), then the time for solving the system is

$$(4tn \log p)^2 \sim e^{(2+o(1))(m_1 \log p)}.$$

Thus, the asymptotic running time of the first stage is given by

$$e^{(1+o(1))(m_1 \log p + \frac{n}{m_1} \log \frac{n}{m_1})} + e^{(2+o(1))(m_1 \log p)}. \tag{3}$$

We note here that the computation of $g^s \pmod f$ is done by repeated squaring, and takes time polynomial in n and $\log p$. Moreover, the factorization of the polynomials can be done in probabilistic polynomial time (see [10] for a recent survey on the topic). Those computations introduce a multiplicative polynomial factor in the above estimate, which is absorbed in the $o(1)$ of the exponent.

Let us consider now $m_1 = kn^\alpha (\log n)^\beta$, for some positive constants α, β, k to be determined later. The first exponent in the above expression becomes

$$(1+o(1))\left(kn^\alpha (\log n)^\beta \log p + \frac{1}{k} n^{1-\alpha} (\log n)^{-\beta} \log\left(\frac{1}{k} n^{1-\alpha} (\log n)^{-\beta}\right)\right),$$

while the second exponent is

$$(2+o(1))kn^\alpha (\log n)^\beta \log p.$$

The expressions are minimized for $\alpha = \beta = \frac{1}{2}$. After the substitutions, the expressions become

$$(1+o(1))\left(k \log p + \frac{1}{2k}\right)\sqrt{n \log n},$$

and

$$(1+o(1))\left(2k\log p\sqrt{n\log n}\right).$$

We view the multiplicative constant $k\log p+\frac{1}{2k}$ as a function of k, and observe that it achieves a minimum at $k=(2\log p)^{-1/2}$. This minimum is $\sqrt{2\log p}$. Substituting everything in Equation (3), we obtain an upper bound for the running time of the first stage

$$L(n)^{\sqrt{2\log p}}=e^{\left(\sqrt{2\log p}+o(1)\right)\sqrt{n\log n}}.$$

Now that the parameters have been fixed, we can easily compute an upper bound for the second stage. This is the expected number of repetitions until the first success. Therefore, an upper bound is

$$\begin{aligned} e^{(1+o(1))m_1\log p} &= e^{(1+o(1))\log p(2\log p)^{-1/2}\sqrt{n\log n}} \\ &= e^{\left(\sqrt{\log p/2}+o(1)\right)\sqrt{n\log n}}=L(n)^{\sqrt{\log p/2}}. \end{aligned}$$

Clearly, the running time of the algorithm is dominated by the first stage, and an upper bound for it is, as $n\to\infty$,

$$L(n)^{\sqrt{2\log p}}.$$

□

Remark. The algorithm as described in Section 2 works with any factor base S. The subsequent analysis shows that if the factor base S consists of all monic irreducible polynomials over $\mathbb{F}_p$ with degree greater than m_2 and less than or equal to $\sqrt{n\log n/(2\log p)}$, for *any* $0\le m_2\le c\sqrt{n\log n/(2\log p)}$ then the algorithm has running time $L(n)^{\sqrt{2\log p}}$. Indeed, the size of the factor base and the probability of success depend only on m_1 — in fact they also depend on m_2, but this dependence is very weak, and is hidden in the $o(1)$ in the exponent. It is clear now, that if we choose the factor base to consist of the set of all monic irreducibles having degree in a set T, then the running time will remain the same provided that T contains all integers from $c\sqrt{n\log n/(2\log p)}$ to $\sqrt{n\log n/(2\log p)}$, for any constant $c<1$. In other words, T does not necessarily have to contain consecutive integers, provided that the above condition is met. The situation is shown in Figure 1.

4. The Waterloo Variant

We now turn to a variant of the basic method in Section 2, that was proposed by Blake, Fuji-Hara, Mullin, and Vanstone [1], known as the Waterloo variant. It is one of the first variants that appeared in the literature, and attempts to improve the running time of the method at the cost of not being able to analyze the variant rigorously. In this section, we show that the same heuristic arguments go through for the generalized factor base.

The basic idea of the Waterloo variant is to find two polynomials w_1 and w_2 of degree at most $n/2$ each, such that

$$w_1h\equiv w_2\pmod f,$$

and try to factor them over the factor base S. Here, f and h are as in the algorithm of Section 2. The polynomials w_1 and w_2 can be computed easily using the extended Euclidean algorithm on input (h,f). More details about the method are given in the original paper [1], and in the survey article by Odlyzko [16].

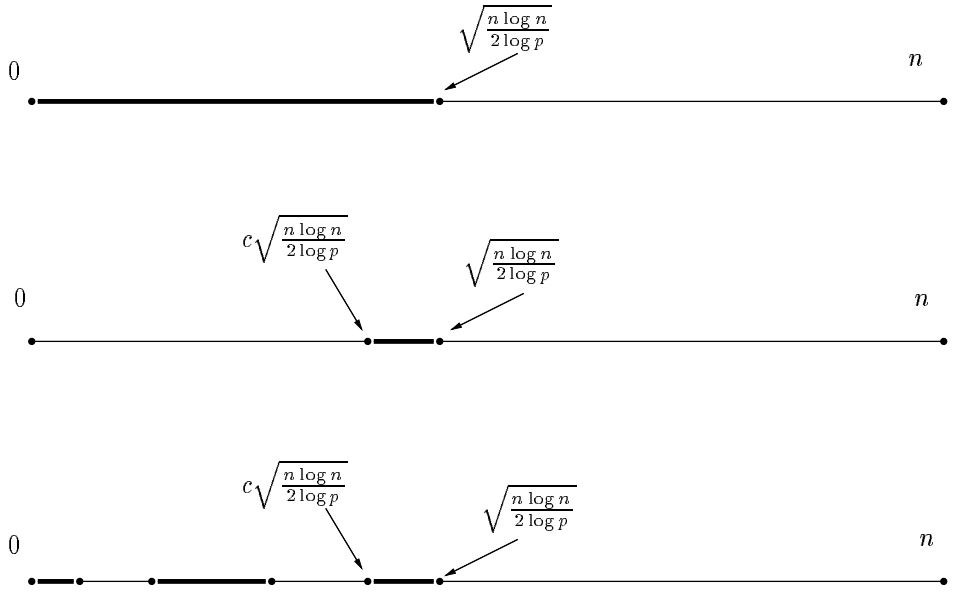

FIGURE 1. The form of the factor base

The problem for a rigorous analysis is that not much is known about the joint distribution of (w_1, w_2). The heuristic assumption is that w_1 and w_2 behave as random *independent* polynomials of degree at most $n/2$. The probability that two random monic independent polynomials of degree at most $n/2$ factor into irreducibles of degree greater than m_2 and at most m_1 is

$$e^{-2(1+o(1))\frac{n}{2m_1}\log\frac{n}{2m_1}} = e^{-(1+o(1))\frac{n}{m_1}\log\frac{n}{2m_1}}. \tag{4}$$

The size of the factor base remains the same as in the basic version, therefore an upper bound on the running time for the first stage is

$$e^{(1+o(1))\left(m_1 \log p + \frac{n}{m_1}\log\frac{n}{2m_1}\right)} + e^{(2+o(1))m_1 \log p}, \tag{5}$$

where we consider the same algorithms and costs as in Theorem 3. Again we let $m_1 = k\sqrt{n \log n}$, and Equation (5) becomes

$$e^{(1+o(1))\left(k \log p + \frac{1}{2k}\right)\sqrt{n \log n}} + e^{(1+o(1))(2k \log p)\sqrt{n \log n}}.$$

The above expression is minimized for $k = (2 \log p)^{-1/2}$, and the running time of the first stage is bounded by

$$L(n)^{\sqrt{2 \log p}}.$$

For this choice of parameters, it is easy to see that the second stage has running time bounded by

$$L(n)^{\sqrt{\log p/2}}.$$

Therefore, the running time of the Waterloo variant is bounded by

$$L(n)^{\sqrt{2 \log p}}.$$

Remark. The practical improvement that this variant provides is hidden in the above analysis in the $o(1)$ term of the exponent. To see the actual improvement

one should compare Equation (3) with Equation (5), and in particular the time to create the system of congruences (the two systems are expected to have the same size, and therefore the time to solve it is expected to be of the same order). The Waterloo variant is faster by a factor of $2^{n/m_1}$, as was the case when the standard base was used (see [16], pp. 238-243).

5. The Coppersmith Variant

In this section, we briefly consider the variant proposed by Coppersmith in [4]. It is designed to work in finite fields of even characteristic, and it relies on unproven assumptions, which however, seem to be reasonable (at least in practice). We note that this was the first method to be conjectured to have running time of the form

$$M(n) = e^{(c+o(1))n^{1/3}(\log n)^{2/3}},$$

where c is an effectively computable constant.

The method assumes that the monic irreducible polynomial f of degree n used to define the field $\mathbb{F}_{2^n}$ is of the form $f(x) = x^n + f_1(x)$, where $\deg f_1 \leq \log_2 n$. Little is known about the irreducibility of this type of polynomials; see [7] for general computational experiments with sparse polynomials over $\mathbb{F}_2$ and for applications of these polynomials. There exist some values of n with no irreducible polynomial of this form. On the other hand, if we let $\deg f_1 \leq \log_2 n + c_1$, for c_1 a small positive integer, there are irreducible polynomials of this form of degree n for all practical values of n (see [7]).

In the first stage of the algorithm, two polynomials w_1 and w_2 are constructed such that

$$w_2 \equiv w_1^{2^k} \pmod{f}, \tag{6}$$

where k is a parameter chosen so that 2^k is on the order of $n^{1/3}(\log n)^{-1/3}$. The construction of w_1 and w_2 is based on algebraic manipulations that are independent of the factor base. Next, the heuristic assumption is made that w_1 and w_2 behave like random *independent* polynomials of degree at most on the order of $n^{2/3}(\log n)^{1/3}$. The analysis follows by essentially the same arguments given in the previous sections. The analysis of the second stage is carried out in a similar manner. For a detailed description of the algorithm, the reader is referred to the original paper by Coppersmith [4], and to the excellent survey by Odlyzko [16].

6. Experimental Results

In this section we present some experimental results for the quantities of interest for the analysis of the basic index calculus method. We fix the degree of the field extension to $n = 500$, and consider three different values for p, namely 2, 3, and 5. Having fixed the values for n and p, we compute the corresponding value for m_1 as determined in Section 3.3, that is

$$m_1 = \sqrt{\frac{n \log n}{2 \log p}}.$$

We consider several possible values for m_2, since this is a "free parameter". The value $m_2 = 0$ corresponds to the standard method with the smooth factor base. For each value of m_2 we compute the number of polynomials of interest $N_p(n, m_1, m_2)$,

the probability of success computed as $p^{-n}N_p(n, m_1, m_2)$, and the size $\#S$ of the factor base.

m_2	$N(n, m_1, m_2)$	Prob	#S
0	.1240e140	.1240e-10	.6125243528e13
1	.2613e139	.2613e-11	.6125243528e13
4	.1001e139	.1001e-11	.6125243528e13
8	.3755e138	.3755e-12	.6125243528e13
12	.1579e138	.1579e-12	.6125243527e13
15	.8222e137	.8222e-13	.6125243523e13
20	.2522e137	.2522e-13	.6125243417e13
23	.1116e137	.1116e-13	.6125242762e13

TABLE 1. $n = 500$, $p = 2$, $m_1 = 47$.

m_2	$N(n, m_1, m_2)$	Prob	#S
0	.1975e224	.1975e-15	.5406460513e17
1	.2471e223	.2471e-16	.5406460513e17
3	.1049e223	.1049e-16	.5406460513e17
5	.5347e222	.5347e-17	.5406460513e17
8	.2220e222	.2220e-17	.5406460513e17
15	.2742e221	.2742e-18	.5406460512e17
18	.9309e220	.9309e-19	.5406460509e17

TABLE 2. $n = 500$, $p = 3$, $m_1 = 38$.

m_2	$N(n, m_1, m_2)$	Prob	#S
0	.1297e331	.1297e-18	.1893590989e21
1	.1473e329	.1473e-20	.1893590989e21
3	.5304e328	.5304e-21	.1893590989e21
5	.2369e328	.2369e-21	.1893590989e21
8	.7794e327	.7794e-22	.1893590989e21
12	.1561e327	.1561e-22	.1893590989e21
15	.3549e326	.3549e-23	.1893590989e21

TABLE 3. $n = 500$, $p = 5$, $m_1 = 31$.

About the finite fields of the experiments, we comment that in Table 1 the size of the field is $2^{500} \approx 10^{150}$, which is on the borderline of what is currently computable. The finite fields in the Tables 2 and 3 are of size $3^{500} \approx 10^{239}$ and $5^{500} \approx 10^{349}$ respectively, which is well beyond the size of the fields for which discrete logarithms are currently computable.

The experiments indicate that for moderately large (but quite reasonable for practical purposes) values for n, the probability of success drops faster than the size of the factor base. This suggests that the common belief that the smooth factor base is "optimal" is, practically speaking, justified.

7. Conclusions

In this paper we propose a variant of the index calculus method using a factor base formed by all monic irreducible polynomials with degree in an interval $(m_2, m_1]$. We prove that the best possible value for m_1 is $(2\log p)^{-1/2}\sqrt{n\log n}$, and then, we have the same asymptotic result as in the standard case, that is,

$$L(n)^{\sqrt{2\log p}} = \exp\left((\sqrt{2\log p} + o(1))\sqrt{n\log n}\right).$$

We also indicate, experimentally, the influence of the parameter m_2. In practical terms, there is a tradeoff associated with m_2. Indeed, smaller values of m_2 imply higher probabilities of success (until a useful congruence is found in the first stage of the algorithm) but the space used in the factor base and the system of congruences to be solved are bigger. On the other hand, larger values of m_2 mean lower probabilities of success but smaller size of factor base and size of the system of congruences to be solved.

The generalized factor base also applies to the heuristic variants described in Sections 4 and 5. Moreover, Odlyzko's survey [16] describes other variants that improve upon the basic method, although not as dramatically as the variant by Coppersmith. All those algorithms use clever algebraic manipulations to compute polynomials of "low" degree, which are subsequently factored. The point to be stressed here is that the computation of the above polynomials is completely independent of the factor base. Therefore, a different factor base like the one proposed in this paper is "compatible" with all the variants. Furthermore, since the probability that a polynomial of degree k factors into irreducibles of degree greater than m_2 and less than or equal to m_1 is of the same form for a wide range of values for m_2, it follows that (at least asymptotically) the variants have the same running time, independently of the factor base used.

References

[1] I.F. Blake, R. Fuji-Hara, R.C. Mullin, and S.A. Vanstone. Computing discrete logarithms in finite fields of characteristic two. *SIAM J. Alg. Disc. Meth.*, 5:276–285, 1984.

[2] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM J. Comput.*, 13:850–864, 1984.

[3] J. Buchmann and S. Düllmann. On the computation of discrete logarithms in class groups. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 134–139. Springer, 1991.

[4] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Info. Theory*, 30:587–594, 1984.

[5] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22:644–654, 1976.

[6] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Info. Theory*, 31:469–472, 1985.

[7] S. Gao, J. Howell, and D. Panario. Irreducible polynomials of given forms. In R.C. Mullin and G.L. Mullen, editors, *Finite fields: theory, applications and algorithms*, volume 225, pages 43–54. Contemporary Mathematics, Amer. Math. Soc., 1999.

[8] S. Gao, J. von zur Gathen, and D. Panario. Gauss periods: orders and cryptographical applications. *Math. Comp.*, 67:343–352, 1998.

[9] T. Garefalakis and D. Panario. Polynomials over finite fields free from large and small degree irreducible factors. Submitted, 18 pages, 1999.

[10] J. von zur Gathen and D. Panario. A survey on factoring polynomials over finite fields. To appear in *J. Symb. Comp.*, 2000.

[11] R. Lidl and H. Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1983.

[12] R. Lovorn. *Rigourous, subexponential algorithms for discrete logarithms over finite fields.* PhD thesis, University of Georgia, 1992.

[13] R. Lovorn Bender and C. Pomerance. Rigourous discrete logarithm computations in finite fields via smooth polynomials. In *Computational Perspectives on Number Theory Proc. of a Conference in Honor of A.O.L. Atkin*, volume 7 of *AMS/International Press Studies in Advanced Mathematics*, Providence, 1998.

[14] A.J. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, Boston, Dordrecht, Lancaster, 1993.

[15] V. Müller, A. Stein, and C. Thiel. Computing discrete logarithms in real quadratic congruence function fields of large genus. *Math. Comp.*, 68:807–822, 1999.

[16] A. Odlyzko. Discrete logarithms and their cryptographic significance. In *Advances in Cryptology, Proceedings of Eurocrypt 1984*, volume 209 of *Lecture Notes in Computer Science*, pages 224–314. Springer-Verlag, 1985.

[17] A. Odlyzko. Discrete logarithms and smooth polynomials. In G.L. Mullen and P. J.-S. Shiue, editors, *Finite fields: theory, applications and algorithms*, pages 269–278. Contemporary Mathematics 168, Amer. Math. Soc., 1994.

[18] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory*, 32:54–62, 1986.

Department of Computer Science, University of Toronto, Toronto, M5S 3G4, Canada
E-mail address: theo,daniel@cs.toronto.edu