rijksuniversiteit groningen

faculteit Wiskunde en Natuurwetenschappen

Discrete logarithm problems in finite fields and their applications in cryptography

Master's Thesis Mathematics

May 2015

Student: M.R. Dam

First Supervisor: Prof.dr. J. Top

Second Supervisor: Prof.dr.ir. R.W.C.P. Verstappen

Abstract

The discrete logarithm problem in finite fields consists of, given two elements a and b in the finite field, solving $a^k = b$ for k. When the size of the finite field is large, solving a discrete logarithm is assumed to be very difficult. The security of several cryptographic systems is based on the fact that this problem cannot be solved in polynomial time. As it is used in cryptographic systems, it has been intensively studied over the last decades. The most efficient algorithms nowadays to solve discrete logarithm problems are the index calculus algorithms. In this thesis, the number field sieve, the function field sieve and a new index calculus algorithm reaching quasi-polynomial complexity are studied.

Contents

Chapter 1

Introduction

The discrete logarithm problem in finite fields consists of, given a finite field $\mathbb{F}_Q$ of size a prime or prime power Q and two elements $a, b \in \mathbb{F}_Q$, finding an integer k such that

$$a^k = b \qquad \text{in } \mathbb{F}_Q$$

This problem is known to be very difficult, or even infeasible, as Q gets large. The security of several cryptographic systems is based on the fact that this problem cannot be solved in polynomial time. The discrete logarithm is, together with factorization of large integers, the major pillar in public key cryptography. Indeed, protocols such as the Diffie-Hellman key exchange and the Digital Signature Algorithm are based on the assumption that the discrete logarithm problems that appear in it are unsolvable. In chapter 2 these protocols are discussed.

As it is used in cryptographic systems, it has been intensively studied over the last decades. Several methods are developed which are optimal for different Q (e.g. Q prime or $Q = p^n$ with p prime), where the index calculus algorithms are lately the most efficient ones. Hence, they are the main focus of this thesis. In chapter 3 the problem and the general principle of index calculus algorithms will be treated.

When using these algorithms, it is an interesting question how long it will take before the algorithm terminates, i.e. how many primitive operations are needed. This will always be a heuristic analysis based on several assumptions that have to be made along the way, but such a complexity analysis gives a measure for the complexity (or the running time) in terms of the useful $L_Q[s, c]$-function, with which it is easy to compare algorithms. This is explained in chapter 4.

Nowadays, there are three main types of index calculus algorithms. The first two, the number field sieve (NFS, discussed in chapter 5) and the function field sieve (FFS, chapter 6) are sieving methods, but the finite fields for which they work optimal are different. The NFS works well in medium to large characteristic, while the FFS is optimal in small characteristic. As

the name suggests, they make use of number fields and function fields, respectively, and some theory in this will be treated. After a description and complexity analysis of the algorithm, an example is given, followed by some results of how it works in practice.
The third and most recent index calculus algorithm, published in 2013 and here named the new index calculus algorithm, works for fields of small characteristic and gives spectacular bounds in the complexity analysis, because sieving (a bottleneck of the NFS and FFS) is no longer necessary. This method is discussed in chapter 7, together with an example, latest adaptions and open questions regarding the discrete logarithm problem in small characteristic.

Chapter 2

Applications of discrete logarithms

In this chapter, two applications of discrete logarithms in cryptography are described. Both are based on the assumption that the discrete logarithm problems that appear in it are too hard to solve for an attacker. The first is the earliest example of a method using a public key, namely the Diffie-Hellman key exchange, which is a method to safely send a message over an insecure network. The second example is a digital signature scheme, used to sign a document such that the reader can verify that it was written by the sender.

2.1 Diffie-Hellman key exchange

Suppose Alice wants to send a message to Bob safely, but the only way of transporting the message is via an insecure network. A group of algorithms to do this with is the so-called *public key cryptography*, which makes use of a key that is public, so known to everyone and in particular to the attacker Eve. Another key, the so-called *private* key, is known only by the mailer (Alice) and the receiver (Bob). The idea of public key cryptography came from Diffie and Hellman in 1976 ([DH76]). Assuming that we are working in a finite field $\mathbb{F}_p$, the Diffie-Hellman key exchange works as follows:

- Alice and Bob agree on a generator $g \in \mathbb{F}_p$. It is assumed to be public, so Eve knows g as well.
- Alice picks an $a \in \mathbb{F}_p$ and sends $A = g^a$ to Bob.
- Bob picks a $b \in \mathbb{F}_p$ and sends $B = g^b$ to Alice.
- Alice computes $B^a = (g^b)^a$, Bob computes $A^b = (g^a)^b$, so they both end up with the same number.

Note that the values of A and B are sent via the insecure information channel, so both are known to Eve. Eve knows g, A and B, so she could find a or b only[1] if she could solve $g^a \equiv A \mod p$ or $g^b \equiv B \mod p$ for a and b (in this section mod denotes the binary operation: division with remainder), which are discrete logarithm problems.
Suppose the key is exchanged, i.e. Bob has received A and Alice has received B. Now Alice can send a message m to Bob as mg^{ab}. Bob now only has to compute $(g^{ab})^{-1}$ to decrypt the message. As he knows b (and knows that the order of $\mathbb{F}_p$ is $|\mathbb{F}_p| = p - 1$), this is done as follows:

$$(g^a)^{|\mathbb{F}_p|-b} = g^{a|\mathbb{F}_p|-ab} = 1^a g^{-ab} = (g^{ab})^{-1}.$$

Hence, the message m is sent safely from Alice to Bob, assuming that Eve cannot compute a and b from $g^a \equiv A \mod p$ or $g^b \equiv B \mod p$. Recent guidelines[2] suggest that the prime p should be $p \approx 2^{1000}$ and $g \approx p/2$ for the discrete logarithm problem to be hard enough so that Eve cannot solve it.

2.2 Digital signature algorithm

A digital signature is in a way a modern version of a signet ring, a ring with which the owner can seal an envelope with some candle wax. Only the owner can make this seal, while everybody can confirm that the letter was sealed by (and hence written by or approved by) the owner of the ring. In the same way a digital signature can be used to check that a computer file was indeed made or approved by the sender. This is important, for example, if your computer wants to download an update. How can you confirm that this is indeed made by a legitimate source and not by a hacker? One way to do this is to use the Digital Signature Algorithm (DSA), which will be discussed in this section. This method is developed by the U.S. government in 1991 and is still widely used.
There are three steps in the DSA. Suppose Sam the Sender wants to send a signed document to Victor the Verifier. First, two keys are created by Sam: a private signing key and a public verification key. Next, Sam signs the document using the signing key. Then, the receiver Victor performs the verification steps using the verification key to see if Sam indeed signed the document. Note that this verification algorithm only has access to the public key, *not* to the private key used to sign the document. We work in a subgroup of $\mathbb{F}_p^*$ of prime order q. The three steps and the task an attacker faces when he wants to forge Sam's signature are explained below.

[1] As far as is known, this is the only way for Eve to find the secret value without help from Alice or Bob ([HPS08]).
[2] Guidelines of 2008, given in [HPS08].

Creating the key. Sam chooses two primes p and q such that $p \equiv 1 \mod q$ and fixes a $g \in \mathbb{F}_p$ of order q. This can easily be done, for example, take $g = g_1^{(p-1)/q}$ for a primitive root g_1 of $\mathbb{F}_p$. Then, she chooses a secret exponent s and computes $v \equiv g^s \mod p$. Note that p, q and g are public knowledge, so the total public verification key is (v, p, q, g).

Signing the document. The document D is a number $1 \leq D < q$. Now, Sam chooses an ephemeral key e satisfying $1 \leq e < q$ and computes:

$$S_1 \equiv (g^e \mod p) \mod q \qquad S_2 \equiv (D + sS_1)e^{-1} \mod q.$$

She sends (S_1, S_2) to Victor together with the document D.

Verification. Victor computes:

$$V_1 \equiv DS_2^{-1} \mod q \qquad V_2 \equiv S_1 S_2^{-1} \mod q$$

and checks if $(g^{V_1} v^{V_2} \mod p) \mod q$ equals S_1. If no attacker is involved, by simply substituting all the variables it follows that this should be true. If this is the case, the algorithm returns True, so the signature is valid. Else, it returns False and the document is probably not sent by Sam.

Attacking. If an attacker, who knows g, v and p, can solve $g^s \equiv v \mod p$ then he can forge Sam's signature. Since this is a discrete logarithm problem this is assumed to be infeasible when p and q are of a good size. Trying to recover s from the rest of the steps is (as far as is known) at least equally difficult ([HPS08]).

Chapter 3

The problem

The discrete logarithm problem in $\mathbb{F}_Q$, a finite field of size Q consists of, given $a, b \in \mathbb{F}_Q$, finding an integer $0 \leq k \leq Q-1$ such that

$$a^k = b \qquad \text{in } \mathbb{F}_Q.$$

The integer a is called the *base*. The discrete logarithm of b to the base a is denoted by $\log_a b$. If a is a primitive root in $\mathbb{F}_Q$, such a solution always exists. If a is not a primitive root, there may be no solution. In this thesis, assume a is a primitive root in $\mathbb{F}_Q$.
The logarithm $\log_a b$ is actually defined modulo $(Q-1)$, since $a^{Q-1} \equiv 1 \mod Q$, hence if k is a solution then $a^{k+c(Q-1)} \equiv b \mod Q$ as well for any constant $c \in \mathbb{Z}$. Usually, $\log_a b$ denotes the value $0 \leq \log_a b \leq Q-2$. Note that for the discrete logarithm we have the familiar-looking computation rules:

$$\log_a(bc) \equiv \log_a(b) + \log_a(c) \mod (Q-1) \tag{3.1}$$
$$\log_a(b^s) \equiv s \log_a(b) \mod (Q-1). \tag{3.2}$$

The most naive way of solving a discrete logarithm would be trial and error, but $k \in \{1, \ldots, Q-1\}$, so this can almost literally take forever when this is done for a large Q. Hence, tricks were applied to speed up the computation. For example, two algorithms are the *Pohlig-Hellman algorithm* (1978) or *Shank's baby-step-giant-step algorithm* (1971) (see e.g. [HPS08] for an introduction). Nowadays, the main techniques for solving a discrete logarithm are the *index calculus algorithms* since they are most efficient ones.

3.1 Index calculus algorithms

We focus in this thesis on the so-called index calculus algorithms. First, the following definition will be useful:

Definition 3.1. A number is called *smooth* if it factors completely into small prime numbers. It is called B-smooth if none of its prime factors is greater than B. B is called the *smoothness bound*, the set $\{p : p < B, p \text{ prime}\}$ is called the *factor base*.

The index calculus algorithm always works in three steps:

1. Collect linear relations involving elements of the factor base.
2. Solve the linear relations using linear algebra.
3. Solve for the individual logarithm.

Sometimes, solving for the individual logarithm is done in the linear algebra step. Here, a straightforward index calculus algorithm will be considered. It is not a very efficient one, but it nicely illustrates the three steps. For simplicity, assume the problem has the form $a^k \equiv b \mod p$, with p prime and a a primitive root in $\mathbb{F}_p$. We are looking for powers of a which factor into small enough primes.

The first step is to fix a smoothness bound B, and list all primes $p_1, \ldots, p_n \leq B$. Now, find relations by exponentiating a to some integer power $r > 0$ and checking if the result is B-smooth. If it is, then this yields a relation of the form:

$$\prod_j p_j^{r_j} \equiv a^r \mod p$$

where all $p_j \leq B$. Rewrite this as:

$$\prod_j a^{\log_a(p_j) r_j} \equiv a^r \mod p$$

Take the logarithm to get a linear relation on indices:

$$\sum_j r_j \log_a(p_j) \equiv r \mod p - 1$$

When you have enough of such equations (i.e. there are n linear independent ones), the linear relations can be put in matrix form, where the $\log_a(p_j)$ are the unknowns. Here, we arrive at the second step, the linear algebra step. We have to solve the matrix equation found in the first step. When this is solved, we know the discrete logarithm of all primes $p_1, \ldots, p_n \leq B$.
The last step consists of computing the individual logarithm of the element b. If b can be factored in primes $p_1, \ldots, p_n \leq B$, then, using equation (3.1), the discrete logarithm is found. If b is not B-smooth, tricks can be used. For example, one can try random l until $a^l b \mod p$ is B-smooth (which most likely will happen for some l). As $\log_a(a^l) = l$, $\log_a b$ is still easily computed.

This already shows that the smoothness bound B must be chosen carefully: not too large, as you will have to solve a large matrix, but also not too small, as the chance of finding a relation will decrease.
As said, the method above is not an efficient one. The now known optimal method depends on the finite field you work in. It can be summarized as follows:

- $\mathbb{F}_p$ or $\mathbb{F}_{p^n}$ where p is a medium to large prime: number field sieve.
- $\mathbb{F}_{p^n}$, p a small prime prime: function field sieve.
- $\mathbb{F}_{q^{2k}}$, where $q = p^n$, $q \approx k$ in small characteristic: new index calculus algorithm.

The ideas of these three algorithms are the same: you first create relations between elements in the factor base and then solve these relations. The set-up is however very different, as the names suggest. In the next sections, these methods are explained. The linear algebra step, which consists of matrix equation involving a large but sparse matrix, can be done using several methods, like the Lanczos algorithm or structured Gaussian elimination. This is not our focus, so when needed, we assume standard methods are used taking a reasonable amount of time to solve it.

3.2 Records

The current records (as of 2014) of discrete logarithms in finite fields are listed below. The records denote the largest finite field in which a discrete logarithm was solved.

- In a finite field of characteristic 2, the current record is in $\mathbb{F}_{2^{9234}} = \mathbb{F}_{2^{9 \cdot 2 \cdot 513}}$ (a twisted Kummer extension) using the new index calculus algorithm ([ZGK14]). The advantage of such an extension is explained in 7.4.
- In a finite field of characteristic 2 of prime degree, the current record is in $\mathbb{F}_{2^{809}}$ using the function field sieve ([BBD+14]).
- In a finite field of characteristic 3, the current record is in $\mathbb{F}_{3^{2395}}$, using the new index calculus algorithm ([JP14]).
- In a finite field of medium characteristic, notable computations are in $\mathbb{F}_{65537^{25}}$ using the function field sieve ([JL06]), and in $\mathbb{F}_{3334135^{357}}$ using pinpointing, as explained in section 6.7 ([Jou13]).

3.3 Sieving

Both the number field sieve and the function field sieve that will be discussed are so-called *sieving* methods. The best known sieve is the sieve of Eratosthenes, which is a simple algorithm for finding all prime numbers up to a given limit n. It works as follows:

1. Create a list of the integers from 2 through n: $(2, 3, 4, \ldots, n)$.
2. Initially, let $p = 2$, which is the first prime number.
3. Mark all multiples of $p \leq n$ (so $2p, 3p, 4p, \ldots$), but not p itself.
4. Find the first number greater than p which is not marked. If $p > \sqrt{n}$, stop. Otherwise, let p equal this new number. Now, repeat from step 3.

When the algorithm terminates, all unmarked numbers are the prime numbers: they are not a product of any other numbers.
The same idea is applied in the sieving algorithms. We will be searching for objects that factor into factors that are contained in the factor base. We do this using a polynomial sieve as described in [Sch93]. First, we discuss the sieve in the case of a polynomial with one variable. Let $f \in \mathbb{Z}[x]$ be a polynomial of degree d, B a given smoothness bound and $C > 0$ a constant. List the B-smooth primes as $p_1, \ldots, p_n$. We want to find all values $a \in [-\frac{1}{2}C, \frac{1}{2}C]$ for which $f(a)$ is B-smooth.
For the sieve we need the following principle: we can find all values of a in the interval $[-\frac{1}{2}(q_i - 1), \frac{1}{2}(q_i + 1)]$ for which q_i divides $f(a)$ by solving $f \bmod q_i$. Since q_i divides $f(a)$ if and only if q_i divides $f(a + q_i)$, we can find all $a \in [-\frac{1}{2}C, \frac{1}{2}C]$ such that q_i divides $f(a)$ by adding $\pm q_i$ to the roots of $f \bmod q_i$.
For all $a \in [-\frac{1}{2}C, \frac{1}{2}C]$, set $d_0(a) = f(a)$. Compute $d_i(a)$ by dividing $d_{i-1}(a)$ by the highest power of q_i dividing it and calling the quotient $d_i(a)$. Whether or not q_i divides it, is checked with the principle described above. Those a for which $f(a)$ is B-smooth are those for which $d_n(a) = \pm 1$.
In case of a homogeneous polynomial of degree d in two variables, so $f \in \mathbb{Z}[x_1, x_2]$ the sieve works as follows. We want to find all values (a_1, a_2) with $a_1, a_2 \in [-\frac{1}{2}C, \frac{1}{2}C]$ such that $f(a_1, a_2)$ is B-smooth. To do this, create $\bar{f}(y)$ by dividing f by x_2^d and let $y = x_1/x_2$, then $\bar{f} \in \mathbb{Z}[y]$ and we can find all $b \in [-\frac{1}{2}C, \frac{1}{2}C]$ such that q_i divides $\bar{f}(b)$ with the sieve for polynomials in one variable. For each $a_2 \in [-\frac{1}{2}C, \frac{1}{2}C]$, we obtain values $f(a_1, a_2)$ divisible by q_i by finding $a_1 \in [-\frac{1}{2}C, \frac{1}{2}C]$ such that $a_1 \equiv ba_2 \mod q_i$. Then, the $d_n((a_1, a_2))$ are computed in the same way as in the one variable case. Again, the values (a_1, a_2) with $a_1, a_2 \in [-\frac{1}{2}C, \frac{1}{2}C]$ such that $f(a_1, a_2)$ is B-smooth are the values for which $d_n((a_1, a_2)) = \pm 1$.
Although complexity analysis is not explained until chapter 4, we analyze

the time complexity of this sieve here as we need the result later on. We can find the roots of $\bar{f}$ over $\mathbb{Z}/q_i\mathbb{Z}$ in time bounded by $(d + \log q_i)^{O(1)}$ ([Len90]). The time required to find the (a_1, a_2) for which $f(a_1, a_2)$ is divisible by q_i and to compute $d_i((a_1, a_2))$ for these (a_1, a_2) is $O(C^2/q_i)$. Summing over all q_i, the running time of this sieve is:

$$\pi(B)(d + \log B)^{O(1)} + C^2 \log\log B,$$

where $\pi(B)$ denotes the number of primes less than or equal to B.

Chapter 4

Complexity analysis

In order to compare different algorithms, their *running time* or *complexity* must be analyzed. This is done by approximating the number of primitive operations that is executed during the algorithm. Such analyses are *heuristic*: several assumptions have to be made along the way, making it not an exact analysis. Also, only the *asymptotic* running time is derived: the behaviour of the algorithm is studied when the size of the input elements is assumed to go to infinity.
In this chapter the basics are explained. First, commonly used asymptotic notations are explained in section 4.1. The terms polynomial time and exponential time algorithms are explained in section 4.2. The $L_Q[s, c]$-notation, commonly used in the analyses of factoring and discrete logarithm algorithms, is introduced in chapter 4.3.

4.1 Asymptotic notations

The big-oh notation $O()$ is used to denote an asymptotic upper bound.

Definition 4.1. For two functions $f(x)$ and $g(x)$, $f(x) = O(g(x))$ if there exists a positive constant c and a positive integer x_0 such that $0 \leq f(x) \leq cg(x)$ for all $x \geq x_0$.

Intuitively, $f(x) = O(g(x))$ means that $f(x)$ grows asymptotically no faster than $g(x)$ to within a constant multiple. Another frequently used notation is the $o()$-notation. The definition is as follows:

Definition 4.2. For two functions $f(x)$ and $g(x)$, $f(x) = o(g(x))$ if for any positive constant $c > 0$ there exists a constant $x_0 > 0$ such that $0 \leq f(x) \leq cg(x)$ for all $x \geq x_0$.

Hence $f(x) = o(g(x))$ means that $g(x)$ is an upper bound for $f(x)$ that is not asymptotically tight, so $f(x)$ becomes insignificant relative to $g(x)$ as x gets larger. Usually, the expression $o(1)$ is used to identify a function $f(x)$ which

satisfies $\lim_{x\to\infty} f(x) = 0$. Another useful notation is $\Theta()$, which denotes an asymptotic tight bound:

Definition 4.3. For two functions $f(x)$ and $g(x)$, $f(x) = \Theta(g(x))$ if there exist positive constants c_1 and c_2 and a positive integer x_0 such that $c_1 g(x) \leq f(x) \leq c_2 g(x)$ for all $x \geq x_0$.

4.2 Polynomial and exponential time

The running time of an algorithm on an input size $\log Q$, where the log denotes the natural logarithm[1], is the number of primitive operations that is executed while running the algorithm. Primitive operations are addition, subtraction, multiplication, division and exponentiation. The input size is the total number of bits needed to represent the input in ordinary binary notation.

An algorithm is said to run in *polynomial time* if the worst-case running time function is of the form $O((\log Q)^k)$, where k is a positive constant. A *quasi-polynomial time* algorithm has a worst-case running time of $2^{O((\log\log Q)^k)}$ for some fixed positive constant k. If it has a worst-case running time of the form Q^k for some constant $k > 0$ it is said to run in *exponential time*.

For example, the most naïve algorithm to solve the discrete logarithm problem $a^k \equiv b \mod p$ would be to try each power of a, until a solution is found. As $1 \leq k \leq p-1$, this algorithm is guaranteed to solve the problem in at most $p-1$ operations. Even though the computation of $a^i \mod p$ for one $i \in \{1, \ldots, p-1\}$ can be done in polynomial time, so in time $O(\log p)$, doing this approximately p times makes the algorithm exponential in p.

Polynomial time algorithms are usually considered efficient algorithms, while exponential time algorithms are inefficient. Note, however, that in cryptography, average-case complexity is more important than worst-case complexity, as it is supposed to be difficult to solve in any case, not just in the worst case.

4.3 $L_Q[s, c]$-notation

For analyzing the complexity of algorithms in number theory, the $L_Q[s, c]$-notation is a very useful tool. Let $\log Q$ be the input size of an algorithm, where Q is usually the cardinality of the considered group. $L_Q[s, c]$ is defined as:

$$L_Q[s, c] = \exp(c(\log Q)^s (\log\log Q)^{1-s})$$

where $0 \leq s \leq 1$, and c is a positive constant. Note that for $s = 0$ we have:

$$L_Q[0, c] = \exp(c(\log\log Q)^1) = (\log Q)^c,$$

[1] Throughout this thesis, log will always denote the natural logarithm.

so it is polynomial of degree c in $\log Q$. While for $s = 1$:

$$L_Q[1, c] = \exp(c(\log Q)^1) = Q^c,$$

which is exponential in $\log Q$. If $0 < s < 1$ the algorithm is said to run in *subexponential* time.
The L_Q-function can be seen as a way of averaging $(\log Q)^c$ (polynomial time) and Q^c (exponential time). Clearly, the usual average $((\log Q)^c + Q^c)/2$ makes no sense here because Q^c is much larger, but if we make an average by taking the double logarithm of $L_Q[s, c]$ we get a useful average. Indeed:

$$\log\log L_Q[s, c] = s \log\log(Q^c) + (1 - s) \log\log(\log Q)^c,$$

which is a combination of $(\log Q)^c$ and Q^c, linear in the $\log\log$ of these terms. So, the s-value in $L_Q[s, c]$ says something about how subexponential the algorithm is on a scale from 0 to 1, 0 being polynomial and 1 being exponential. The complexity of discrete logarithm algorithms is presented in this way, sometimes abbreviated to $L[\alpha]$, which makes it easy to compare them: the smaller α, the better the algorithm.
In particular, looking at the $L_Q[s, c]$-function, we see the following computation rules:

$$\begin{aligned} (L_Q[s, c])^a &= L_Q[s, ac] \\ L_Q[s_1, c_1] \cdot L_Q[s_2, c_2] &< L_Q[\max\{s_1, s_2\}, c_1 + c_2]. \end{aligned}$$

The L-notation appears naturally when considering the Dickman-De Bruijn function ([Dic30]). Given a smoothness bound B, let

$$\Psi(x, B) = \#\{r : 1 \leq r \leq x \text{ and } r \text{ is } B\text{-smooth}\}$$

be the number of B-smooth integers less than x. An estimate for this function is given by the Dickman-de Bruijn function. Let $u = \log x / \log B$, then this estimate is given by:

$$\Psi(x, x^{1/u}) \backsim x\rho(u)$$

where $\rho(u)$ is the solution to the equation:

$$u\rho'(u) + \rho(u - 1) = 0,$$

with initial condition $\rho(u) = 1$ $(0 \leq u \leq 1)$. The solution to this equation is estimated as:

$$\rho(u) \backsim u^{-u+o(u)}.$$

Suppose that x is of the form $x = L_Q[s, c]$ and $B = L_Q[s_B, c_B]$, both depending on the input size $\log Q$. Then the probability that a random integer

chosen from the domain $1 \leq r \leq x$ is B-smooth is given by

$$\begin{aligned}
\frac{\Psi(x,B)}{x} &= u^{-u+o(u)} \\
&= \left(\frac{c(\log Q)^s(\log\log Q)^{1-s}}{c_B(\log Q)^{s_B}(\log\log Q)^{1-s_B}}\right)^{-\frac{c(\log Q)^s(\log\log Q)^{1-s}}{c_B(\log Q)^{s_B}(\log\log Q)^{1-s_B}}+o(u)} \\
&= e^{-(s-s_B)\frac{c}{c_B}(\log Q)^{s-s_B}(\log\log Q)^{-s+s_B}(\log\log Q+O(\log\log\log Q))} \\
&= e^{(-(s-s_B)\frac{c}{c_B}+o(1))(\log Q)^{s-s_B}(\log\log Q)^{1-s+s_B}} \\
&= L_Q\left[s-s_B, -(s-s_B)\frac{c}{c_B}+o(1)\right].
\end{aligned}$$

This result can be translated into the following conjecture:

Conjecture 4.4. Let f be a polynomial in n variables over $\mathbb{Z}$ and assume that $|f(x_1,\ldots,x_n)| < A$ whenever all x_i lie in the interval $[-\frac{1}{2}C, \frac{1}{2}C]$ for a constant $C > 0$. Then the probability that $f(a_1,\ldots,a_n)$ is B-smooth for a_i chosen randomly from $[-\frac{1}{2}C, \frac{1}{2}C]$ is $\Psi(A,B)/A$. If B is chosen as $B = L_Q[s_B, c_B]$ and $A = L_Q[s_A, c_A]$, then this probability is:

$$\frac{\Psi(A,B)}{A} = L_Q\left[s_A - s_B, -(s_A-s_B)\frac{c_A}{c_B}+o(1)\right].$$

This conjecture is useful when analyzing the number field sieve in chapter 5.

Chapter 5

The number field sieve

The number field sieve (NFS) is an index calculus algorithm for computing the discrete logarithm over $\mathbb{F}_p$ or $\mathbb{F}_{p^n}$ for p a medium or large prime where $n \ll \log^{1/2} p$. Here, we will consider the method for computing the discrete logarithm over $\mathbb{F}_p$ only. The extension to the algorithm for discrete logarithms over $\mathbb{F}_{p^n}$ for $n > 1$ can be found in [Sch00] and [JL03].
The NFS was originally developed for the factorization of integers ([BLP93]), and was slightly adapted to solve discrete logarithm problems as in turned out to be efficient for these problems as well. As the name suggests, some number theory is involved in this method. This will be introduced first in section 5.1. The algorithm is explained in section 5.2 and its running time is in section 5.3 derived to be $L_p[1/3, (64/9)^{1/3}]$. In section 5.4, an example of the NFS is given. Finally, in section 5.5, some aspects of the algorithm in practice are analyzed, e.g. which polynomial works best to create the number field.

5.1 Algebraic number theory

Here, a short introduction to number fields is given which covers the needed theory for the NFS. For more details, see [Ste12] or [LL93].
A *number field* is a finite field extension of the field $\mathbb{Q}$. When given a monic irreducible polynomial $f(x) = x^\delta + a_{\delta-1}x^{\delta-1} + \cdots + a_0$ of degree δ with coefficients in $\mathbb{Q}$, let $\alpha \in \mathbb{C}$ be a root of this polynomial. Then the field extension $K = \mathbb{Q}(\alpha)$ is a number field, where δ is the *degree of the number field*. The fact that f is irreducible implies that each element in K has a unique expression of the form $\sum_{i=0}^{\delta-1} q_i\alpha^i$, with $q_i \in \mathbb{Q}$. The corresponding *ring of integers* O_K of K is defined as:

$$O_K = \{\beta \in K : \text{ minimum polynomial of } \beta \text{ over } \mathbb{Q} \text{ is element of } \mathbb{Z}[x]\}.$$

Clearly, always $\mathbb{Z} \in O_K$. Also, note that $\mathbb{Z}[\alpha]$ is a subring of K. Each element in $\mathbb{Z}[\alpha]$ can be expressed as $\sum_{i=0}^{\delta-1} s_i\alpha^i$, with $s_i \in \mathbb{Z}$.

The norm $\mathfrak{Na}$ of a non-zero ideal $\mathfrak{a}$ of $\mathbb{Z}[\alpha]$ is defined by $\mathfrak{Na} = \#(\mathbb{Z}[\alpha]/\mathfrak{a})$. A *first degree prime ideal* of $\mathbb{Z}[\alpha]$ is a non-zero ideal $\mathfrak{p}$ of prime norm p. Indeed, this is a prime ideal as $\mathbb{Z}[\alpha]/\mathfrak{p} \cong \mathbb{Z}/p\mathbb{Z}$, which is a field. The set of first degree prime ideals $\mathfrak{p}$ is in bijective correspondence with the set of pairs $(p, c \mod p)$, where p is a prime number and $c \in \mathbb{Z}$ satisfies $f(c) \equiv 0 \mod p$. If $\mathfrak{p}$ corresponds to $(p, c \mod p)$, then $\mathfrak{Np} = p$, the map $\mathbb{Z}[\alpha] \to \mathbb{Z}[\alpha]/\mathfrak{p} \cong \mathbb{Z}/p\mathbb{Z}$ maps α to $(c \mod p)$, and $\mathfrak{p}$ is generated as an ideal by p and $c - \alpha$. The map $\mathbb{Z}[\alpha] \to \mathbb{Z}[\alpha]/\mathfrak{p}$ can be used to test whether a given element of $\mathbb{Z}[\alpha]$ is contained in $\mathfrak{p}$: $\sum_i s_i\alpha^i \in \mathfrak{p}$ if and only if $\sum_i s_i c^i \equiv 0 \mod p$.

If s is a prime number not dividing the index $[O_K : \mathbb{Z}[\alpha]]$, then its factorization in O_K is given by the following proposition (from [Gor93]):

Proposition 5.1. *For a prime number s not dividing the index, suppose f factors in $\mathbb{F}_s[x]$ as*

$$f(x) \equiv \prod_i g_i(x)^{e_i} \mod s$$

with each g_i monic and irreducible mod s, and $g_i \not\equiv g_j$ for $i \neq j$. Then $(s) = \prod_i \mathfrak{s}_i^{e_i}$ for different prime ideals $\mathfrak{s}_i = (s, g_i(\alpha))$ and the norm of each $\mathfrak{s}_i$ is given by $N(\mathfrak{s}_i) = s^{deg(g_i)}$.

We call the prime ideals that lie over a rational prime $\leq B$ dividing the index $[O_K : \mathbb{Z}[\alpha]]$ *bad* prime ideals. These prime ideals do not have a factorization as in proposition 5.1. If the norm of a prime ideal in O_K does not divide the index, the prime ideal is called *good*.
For the NFS, we are interested in B-smooth algebraic integers of the form $c + d\alpha$, where c and d are coprime integers. Such an integer is said to be B-smooth if every prime number dividing its norm is at most B. The norm of $c + d\alpha$ is a function $N : \mathbb{Q}(\alpha) \to \mathbb{Q}$ given by:

$$N(c + d\alpha) = (-d)^\delta f(-c/d),$$

where $f(x)$ is the minimal polynomial of α. From this equation it is clear that

$$|N(\alpha)| = |a_0|,$$

with a_0 the constant term of $f(x)$.

5.2 The algorithm

Given a prime p and $a, b \in \mathbb{F}_p^*$, we try to find an integer k such that:

$$a^k \equiv b \mod p.$$

We assume a and b are B-smooth for some given smoothness bound B and that a is a generator of $\mathbb{F}_p^*$. In short, with the NFS we will find the residue $\log_a b$ modulo a large odd prime divisor l of $p-1$. The algorithm can be modified to compute the residue of $\log_a b$ modulo any prime divisor l of $p-1$, and then with the Chinese remainder theorem, $\log_a b$ can be determined.
Suppose we are given a smoothness bound B and a constant C, which is a bound on the sieving elements. First, we first fix a number field. Then we apply a sieve to find smooth elements. We compute character maps that enable us to construct elements that are l-th powers in the ring of integers of our number field. Then in the linear algebra phase $\log_a b \mod l$ is computed.

Choosing a number field. For the NFS, we need an integer m and an irreducible monic polynomial $f(x) \in \mathbb{Z}[x]$ such that $f(m) \equiv 0 \mod p$. Usually, in order to find a polynomial f with a root modulo p, the base-m method is used: choose an m of size $\lfloor p^{\frac{1}{\delta}} \rfloor$, where δ denotes the degree of the number field, and write p in base m:

$$p = \sum_{i=0}^{\delta} a_i m^i$$

where all $a_i \in \mathbb{Z}_{>0}$ are less than m, and $a_k = 1$. The polynomial $f(x) = x^\delta + a_{\delta-1}x^{\delta-1} + \cdots + a_0$ now clearly satisfies $f(m) \equiv 0 \mod p$. By [BFO81], this f is irreducible.
With this choice of m, f is monic and the coefficients of f are bounded by $p^{1/\delta}$. Furthermore, it is assumed that l does not divide the discriminant of $f(x)$. If this does not hold, the necessary modifications can be found in [Sch93].
Let $\alpha \in \mathbb{C}$ denote a root of f, then $K = \mathbb{Q}(\alpha)$ is the number field we are working in. Let O_K denote the ring of integers of K. Note that with this construction, there exists a ring homomorphism $\phi : \mathbb{Z}[\alpha] \to \mathbb{F}_p$ such that $\phi(\alpha) = m$.

Sieving. We search for the set D of elements $(c, d) \in \mathbb{Z} \times \mathbb{Z}$ such that c and d are coprime, $-\frac{1}{2}C < c < \frac{1}{2}C$, $0 < d < C$ and both $c + d\alpha$ and $c + dm$ are B-smooth. Let $N : \mathbb{Q}(\alpha) \to \mathbb{Q}$ be the norm map. An element $\gamma \in O_K$ is said to be B-smooth if $N(\gamma)$ is B-smooth in $\mathbb{Z}$, or equivalently, if each first degree prime ideal dividing the ideal generated by γ lies over a rational prime $\leq B$.
If a prime $r \leq B$ appears in the factorization of $N(c + d\alpha)$ we still need to check which first degree prime ideal divides the ideal generated by $c + d\alpha$. First, we check which first degree prime ideals lie over a prime $r \leq B$. This is done by checking whether the corresponding principal ideal (r) factors. One of the following will happen: it will not split (it remains prime), or it splits into factors none of which have degree 1, or it splits into factors some

(or all) of which have degree 1. To determine which of those possibilities is the case, first reduce the coefficients of f modulo r. Denote the remaining polynomial f_r and factor $f_r \mod r$. One of the following will happen, which correspond precisely to the possible ways for (r) to split: f_r is irreducible, f_r factors, but all of its irreducible factors have degree larger than 1 or f_r consists of (some) factors of degree 1. Only in the last case, each distinct degree 1 factor corresponds to a first degree prime ideal with r as its norm. These prime ideals are added to the factor base. Note that from this factorization, for each first degree prime ideal $\mathfrak{r}$, the generators can be read off directly. Indeed, if $(x - e)$ is a factor of f_r, then clearly e satisfies $f(e) \equiv 0 \mod r$, so $\mathfrak{r}$ is generated by $(r, e - \alpha)$. This prime ideal divides a particular $c + d\alpha$ if and only if $c + de \equiv 0 \mod r$. So, by simply checking if $c + de \equiv 0 \mod r$, it is clear which prime ideal divides $c + d\alpha$.
We find the set D by using the polynomial sieve as described in 3.3. We can write both $c + dm$ and $c + d\alpha$ as homogeneous polynomials in two variables, namely: $c + dm = g(c, d)$ where $g(X, Y) = X + Ym$ and likewise, $N(c + d\alpha) = \hat{f}(c, d)$ with $\hat{f}(X, Y) = \sum_i^{\delta} a_i X^i (-Y)^{\delta - i}$. Here, a_i are the coefficients of $f(x)$ and δ is the degree of $f(x)$.

Constructing vectors. Let $\pi(B)$ be the number of rational primes $\leq B$ and let $q_1, \ldots, q_{\pi(B)}$ be a list of those primes. For a rational prime q and integer g, let $v_q(g)$ be the exponent to which q divides g. Similarly, let $\mu(B)$ be the number of good first degree prime ideals of O_K and let $\mathfrak{q}_1, \ldots, \mathfrak{q}_{\mu(B)}$ be a list of those ideals. For each prime ideal $\mathfrak{q} \subseteq O_K$ and element $\gamma \in O_K$, let $v_{\mathfrak{q}}(\gamma)$ be the exponent to which $\mathfrak{q}$ divides the ideal generated by γ.
For $(c, d) \in S$, compute the vector $V_{c,d}$ of length $\pi(B) + \mu(B) + \delta$ whose first $\pi(B)$ entries are:

$$v_{q_1}(c + dm), \ldots, v_{q_{\pi(B)}}(c + dm),$$

whose next $\mu(B)$ entries are:

$$v_{\mathfrak{q}_1}(c + d\alpha), \ldots, v_{\mathfrak{q}_{\pi(B)}}(c + d\alpha),$$

and whose last δ entries are the images of $c + d\alpha$ under character maps, as we will define later on. The values $v_{q_i}(c + dm)$ are read off the prime factorization of $c + dm$ and are already obtained during the sieving stage. The same holds for $v_{\mathfrak{q}_i}(c + d\alpha)$ for good prime ideals $\mathfrak{q}_i$.

Computing character maps. We want to be able to construct elements in O_K which are l-th powers. To do so, we need character maps, which are made in the following way. Let:

$$\Gamma = \{\gamma \in O_K \quad : \quad N(\gamma) \neq 0 \mod l\}.$$

Let $\mathfrak{l}$ range over the prime ideals lying above l. Let ϵ be the least common multiple of the orders of the multiplicative groups $(O_K/\mathfrak{l})^*$ times the power

to which $\mathfrak{l}$ appears in the factorization of (l). Since l does not divide the discriminant of f and is therefore unramified in $\mathbb{Q}(\alpha)$, we have for all $\gamma \in \Gamma$:

$$\gamma^\epsilon \equiv 1 \mod l.$$

Let $\lambda : \Gamma \to lO_K/l^2O_K$ be the map sending γ to $(\gamma^\epsilon - 1) + l^2O_K$. We obtain δ maps for each pair $(c, d) \in D$:

$$\lambda_j : \Gamma \to \mathbb{Z}/l\mathbb{Z}$$

by fixing a module basis $\{b_j l + l^2O_K\}_{j=1,\ldots,\delta}$ for lO_K/l^2O_K over $\mathbb{Z}/l\mathbb{Z}$ and projecting λ onto each coordinate. For example, this can be done as follows: consider $r(X) = (c + dX)^\epsilon - 1$ as an element of $(\mathbb{Z}/q^2\mathbb{Z})[X] \mod f(X)$. By the construction of ϵ, all of the coefficients of $r(X)$ will be multiples of q. Therefore, let the character maps $(\lambda_1, \ldots, \lambda_\delta)$ be the d coefficients of $r(X)$, each divided by q. So:

$$r(X) = \lambda_1 q + \lambda_2 qX + \cdots + \lambda_\delta qX^{\delta-1} \in (\mathbb{Z}/q^2\mathbb{Z})[X] \mod f(X)$$

These $(\lambda_1, \ldots, \lambda_\delta)$ are the last δ entries in the vector $V_{c,d}$.
With the linear algebra step, a $\gamma \in \Gamma$ is produced such that $v_\mathfrak{q}(\gamma) \equiv 0 \mod l$ for all prime ideals $\mathfrak{q} \subset O_K$ and such that $\lambda(\gamma) = 0$. Assume that the class number[1] $\#Cl$ of $\mathbb{Q}(\alpha)$ is prime to l, as is expected for large l. Since $v_\mathfrak{q}(\gamma) \equiv 0 \mod l$ for all $\mathfrak{q}$, γ can be written as the l-th power of an ideal $G \in O_K$, so $(\gamma) = G^l$. Now, consider G modulo the set of principal ideals $\subset O_K$. This has order l. Also, it is an element of Cl, hence its order also divides $\#Cl$. In conclusion, the order divides $\gcd(l, \#Cl) = 1$. Thus, we can conclude that G is a principal ideal, so $G = (b)$ for some $b \in O_K$. We see that $(b^l) = (b)^l = (\gamma)$, so $\gamma = b^l \cdot \omega$ for some $\omega \in O_K^*$: γ is the product of an l-th power and a unit $\omega \in O_K^*$.

Any l-th power is mapped to 0 by λ. This can be seen by considering $b^\epsilon \in 1 + lO_K$ for some $b \in \Gamma$, so b^ϵ can be written as $b^\epsilon = 1 + lc$ for some $c \in O_k$. Now, $b^{\epsilon l} = (1 + cl)^l = 1 + l^2c \mod l^3$. But $l^2c \equiv 0 \mod l^2O_K$, so b^l is sent to 0 by λ.
We can now show that ω is mapped to 0 as well, by noting that $\lambda(xy) = \lambda(x) + \lambda(y)$ for $x, y \in O_K$. This can easily be seen if we write $x^\epsilon, y^\epsilon \in 1 + lO_K$ as $x^\epsilon = 1 + al$, $y^\epsilon = 1 + bl$ for $a, b \in O_K$. Then

$$(xy)^\epsilon = (1 + al)(1 + bl) \equiv 1 + (a + b)l \mod l^2.$$

Now, we see:

$$\lambda(xy) = (xy)^\epsilon - 1 \equiv (a + b)l \equiv (x^\epsilon - 1) + (y^\epsilon - 1) \equiv \lambda(x) + \lambda(y) \mod l^2.$$

[1]The class number is defined as the order of the (finite) class group Cl, which is the group of the set of all ideals of O_K modulo the set of all principal ideals of O_K. For more details, see [Ste12].

Hence:

$$0 = \lambda(\gamma) = \lambda(\omega) + \lambda(b^l) = \lambda(\omega) + 0.$$

So, ω is mapped to 0 as well by λ. It is likely in this case that ω, and in turn γ, is an l-th power. For the heuristic argument supporting this, see [Sch93].

Linear algebra. Recall that we are computing $a^k \equiv b \mod l$, where a and b are B-smooth. Let V_a be the vector of length $\pi(B) + \mu(B) + \delta$ whose first $\pi(B)$ entries are $v_{q_1}(a), \ldots, v_{q_{\pi(B)}}(a)$ and whose other entries are zero. V_b is defined similarly, but replace a for b. Now, let A be the matrix whose first column is V_a and remaining columns are the vectors $V_{c,d}$. We have to solve the congruence:

$$AX \equiv -V_b \mod l. \tag{5.1}$$

If V_b is not in the column space of A, increase the parameter C in order to enlarge the set D. It can be expected that the rank of A increases in this way.
Let $(x, \ldots, x_{c,d}, \ldots)$ be a solution of (5.1), indexed by the pairs $(c, d) \in D$. Let

$$\beta = a^x b \prod (c + dm)^{x_{c,d}} \text{ and } \gamma = \prod (c + d\alpha)^{x_{c,d}},$$

where a and b are seen as integers. Now, $v_q(\beta) \equiv 0$ for all primes q, and therefore β is an l-th power of an element in $\mathbb{Z}[\alpha]$. Similarly $v_{\mathfrak{q}} \equiv 0 \mod l$ for all prime ideals $\mathfrak{q} \subseteq O_K$. Also, $\lambda_i(\gamma) = 0$ for $i = 1, \ldots, \delta$. It is now likely that γ is an l-th power in O_K, but it may be not an l-th power in $\mathbb{Z}[\alpha]$. However, since $f'(\alpha)O_K \subseteq \mathbb{Z}[\alpha]$ (for proof, see appendix A), both $f'(\alpha)^l\delta$ and $f'(\alpha)^l\gamma$ are l-th powers of an element in $\mathbb{Z}[\alpha]$. Considering the ring homomorphism $\phi : \mathbb{Z}[\alpha] \to \mathbb{F}_p$ such that $\phi(\alpha) = m$, we see that $\phi(f'(\alpha)^l\beta) = a^x b\phi(f'(\alpha)^l\gamma)$. Hence, $a^x b$ is an l-th power in $\mathbb{F}_p^*$ and we conclude that $x \equiv -\log_a b \mod l$. Once these residues modulo all prime divisors of $p-1$ is known, $\log_a b$ in $\mathbb{F}_p$ can be determined using the Chinese remainder theorem.

5.3 Complexity analysis

The running time analysis consists of several steps. First, the parameters appearing in the problem are expressed as $L_p[s, c]$-functions. Then, the time needed for the sieving and linear algebra are expressed in those parameters. Now it can be determined how many smooth pairs (c, d) are needed and what the maximal size of the sieving elements is. Then, by determining the probability for smoothness, the total complexity can be computed.

Time of sieving and linear algebra. Every parameters will be expressed as an $L_p[s, c]$-function depending on the prime p. Let the smoothness bound B and the degree of the number field δ be given by:

$$B = L_p[s_B, c_B + o(1)] \qquad p^{1/\delta} = L_p[s_\delta, c_\delta + o(1)].$$

So directly, δ is given by:

$$\delta = \frac{1}{c_\delta} + o(1)\left(\frac{\log p}{\log\log p}\right)^{1-s_\delta}.$$

During the sieving stage, we sieve over c and d such that $-\frac{1}{2}C < c < \frac{1}{2}C$ and $0 < d < C$. Define this C by:

$$C = L_p[s_C, c_C + o(1)].$$

As seen in section 3.3, the sieve is expected to take time:

$$\pi(B)(\delta + \log B)^{o(1)} + C^2 \log\log B,$$

which is, after substitution of B and C and ignoring the $\log B$ and $\log\log B$-terms, approximately equal to:

$$\delta \cdot L_p[s_B, c_B + o(1)] + L_p[s_C, 2c_C + o(1)].$$

The linear algebra step, using the Lanczos method ([Tei98]), takes time

$$O(\delta B^2) = \delta \cdot L_p[s_C, c_C + o(1)].$$

Comparing the two expressions for the sieving and linear algebra, we see that the total time required to complete these steps is given by $L_p[s_T, c_T + o(1)]$, where:

$$\begin{aligned} s_T &= \max\{s_B, s_C\}, \\ c_T &= \max\{2c_B, 2c_C\}. \end{aligned}$$

Obtaining enough smooth pairs. To have a reasonable chance of success in the linear algebra step, we need to have found at least $\pi(B) + \mu(B) + \delta$ pairs (c, d) such that $c + dm$ and $c + d\alpha$ are B-smooth during the sieving step. $\pi(B) + \mu(B) + \delta$ is bounded by $\delta B = \delta \cdot L_p[s_B, c_B + o(1)]$.
Let $P_Q = L_p[s_Q, c_Q + o(1)]$ be the probability that an integer $c + dm$ in the domain of the sieve is B-smooth, and $P_R = L_p[s_R, c_R + o(1)]$ the probability that the ideal $(c + d\alpha)$ is B-smooth in the domain of the sieve. The number of pairs checked times the probabilities of being smooth should be greater than the number of smooth pairs needed, hence:

$$C^2 \cdot P_Q \cdot P_R > \delta \cdot B. \tag{5.2}$$

Maximal size of sieving elements. For the maximal size of the sieving elements, using that $|c|, |d| \leq C$, the coefficients of $f(x)$ are bounded by $p^{1/\delta}$ and $m \approx p^{1/\delta}$, we see that $c + dm \approx p^{1/\delta}C$, hence:

$$\begin{aligned} c + dm &< L_p[s_\delta, c_\delta + o(1)] \cdot L_p[s_C, c_C + o(1)] \\ &< L_p[\max\{s_\delta, s_C\}, c_\delta + c_C + o(1)]. \end{aligned}$$

For the norm of the ideal we see:

$$\begin{aligned} N(c+d\alpha) &= (-d)^{\delta} f(-c/d) \\ &< (\delta+1)\cdot L_p[s_\delta, c_\delta + o(1)]\cdot L_p[s_C, c_C + o(1)]^{\delta} \\ &= (\delta+1)\cdot L_p[s_\delta, c_\delta + o(1)]\cdot L_p[s_C + 1 - s_\delta, \frac{c_C}{c_\delta} + o(1)]^{\delta} \\ &< (\delta+1)\cdot L_p[\max\{s_\delta, s_C + 1 - s_\delta\}, c_\delta + \frac{c_C}{c_\delta} + o(1)]. \end{aligned}$$

Probability for smoothness. Using the conjecture (4.4), we can estimate P_Q by considering $\frac{\Psi(c+dm,B)}{c+dm}$. We see:

$$P_Q \geq L_p[\max\{s_\delta, s_C\} - s_B; -(\max\{s_\delta, s_C\} - s_B)\frac{c_\delta + c_C}{c_B} + o(1)].$$

Likewise for P_R we consider $\frac{\Psi(N(c+d\alpha),B)}{N(c+d\alpha)}$ and see:

$$P_R \geq L_p[\max\{s_\delta, s_C+1-s_\delta\}-s_B; -(\max\{s_\delta+1-s_\delta, s_C\}-s_B)\frac{c_\delta^2 + c_C}{c_\delta \cdot c_B}+o(1)].$$

These imply:

$$\begin{aligned} s_Q &\geq \max\{s_\delta, s_C\} - s_B \\ s_R &\geq \max\{s_\delta, s_C + 1 - s_\delta\} - s_B. \end{aligned}$$

Using equation (5.2), to successfully solve the linear system, we need that $C^2 > \frac{\delta B}{P_Q P_R}$, which leads to:

$$\begin{aligned} s_C &\geq \max\{s_B, s_Q, s_\delta\} \\ &\geq \max\{s_B, s_\delta - s_B, s_C + 1 - s_\delta - s_B, s_C - s_B\} \\ &\geq \max\{2s_B, s_\delta, s_C + 1 - s_\delta, s_C\} - s_B. \end{aligned}$$

Minimizing $s_T = \max\{s_B, s_C\}$ subject to the last inequality yields $s_B = s_C = \frac{1}{3}$ and $s_\delta = \frac{2}{3}$. Now, we have:

$$\begin{aligned} P_Q &= L_p\left[\frac{1}{3}; -\frac{1}{3}\frac{c_\delta}{c_B}(1+o(1))\right], \\ P_R &= L_p\left[\frac{1}{3}; -\frac{1}{3}\frac{c_\delta^2 + c_C}{c_\delta c_B}(1+o(1))\right]. \end{aligned}$$

Again using equation (5.2), this yields:

$$2c_C > c_B + \frac{1}{3}\left(\frac{c_\delta}{c_B} + \frac{c_\delta^2 + c_C}{c_\delta c_B}\right).$$

Now, minimizing $c_T = \max\{2c_B, 2c_C\}$ subject to this inequality yields $c_B = c_C = (\frac{8}{9})^{1/3}$ and $c_\delta = (\frac{1}{3})^{1/3}$. Hence, the key parameters are:

$$\begin{aligned} B = C &= L_p\left[\frac{1}{3}; \left(\frac{8}{9}\right)^{1/3} + o(1)\right], \\ \delta &= \left(\frac{(3+o(1))\log p}{\log\log p}\right)^{1/3}. \end{aligned}$$

The total running time of the algorithm is expected to be:

$$L_p[s_T, c_T] = L_p\left[\frac{1}{3}; \left(\frac{64}{9}\right)^{1/3} + o(1)\right].$$

5.4 Example

We want to solve $2^k \equiv 5 \mod 83$. Set $l = 41$ as the prime divisor of $p-1$. Let $m = 8$, then a base m expression of p gives the polynomial $g(x) = x^2 + 2x + 3$. One can check that now holds $g(m) \equiv 0 \mod 83$. Set $\alpha = -1 + \sqrt{2}$ a root of $g(x)$. We set the smoothness bound $B = 5$. The ideals of norm less than or equal to B can be found as follows:

- $g(x) \equiv (x+1)^2 \mod 2$ so only the prime ideal $(2, \alpha+1)$ is of norm 2.
- $g(x) \equiv x(x+2) \mod 3$, so prime ideals of norm 3 are $(3, \alpha)$ and $(3, \alpha+2)$.
- $g(x) \mod 5$ is irreducible, so (5) remains prime.

Say, we pick $c = 0$ and $d = 1$, then $c + dm = 2^3$ and $N(c + d\alpha) = 3$. As $-\frac{c}{d} \equiv 0 \mod 3$, we know that the corresponding prime ideal in this case must be $(3, \alpha)$.
$g(X) \equiv (x+31)\cdot(x+12) \mod 41$, so $(41) = (41, \alpha+31)\cdot(41, \alpha+12)$ and hence $\epsilon = l - 1 = 40$. The character maps for $c = 0$ and $d = 1$ are given by:

$$r(X) = (c + dX)^\epsilon - 1 = 451x + 41 = 11 \cdot lx + 1l.$$

So $(\lambda_1, \lambda_2) = (1, 11)$.
Likewise, for $c = 12$, $d = 11$, we find $c + dm = 2^2 \cdot 5^2$ and $c + d\alpha = 3^5$, and the corresponding prime ideal again turns out to be $(3, \alpha)$. The character maps are $(\lambda_1, \lambda_2) = (5, 14)$.
$\pi(B) = 3$, $\mu(B) = 4$, $\delta = 2$, so the vector $V_{c,d}$ has length 9, but only the primes 2 and 5 and the prime ideal $(3, \alpha)$ appear so we create the vectors $V_{c,d}$ of length 5 with the following entries:

$$V_{c,d} = (\text{power of } 2, \text{power of } 5, \text{power of } (3, \alpha), \lambda_1, \lambda_2).$$

The matrix A consists of the row vectors V_2, $V_{0,1}$ and $V_{11,12}$, where V_2 only contains a 1 on the first entry. So:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 11 & 1 \\ 2 & 2 & 5 & 14 & 5 \end{pmatrix}^T$$

(Of course, normally we would search for more relations, but since this turns out to give out a solution, for brevity we only use two smooth pairs (c, d)). Similarly, V_5 is the row vector with a 1 on the second entry, so:

$$V_5 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T.$$

Now, when we solve $AX = -V_5 \mod 41$, we get as a solution:

$$X = \begin{pmatrix} 14 & 23 & 20 \end{pmatrix}^T.$$

We can check that β is an l-th power in $\mathbb{Z}[\alpha]$:

$$\beta = 2^{14} \cdot 5 \cdot (8)^{23} \cdot (11 + 12 \cdot 8)^{20} = 2^{14} \cdot 5 \cdot (2^3)^{23} \cdot (2^2 \cdot 5^2)^{20} = 2^{123} \cdot 5^{41},$$

and also that γ is an l-th power in O_K:

$$\gamma = (\alpha)^{23}(11 + 12\alpha)^{20} = (3, \alpha)^{123}.$$

For the first entry x in X, we also find that $x = -\log_2 5 = -14 \equiv 27 \mod l$. This is correct, as one can check that $2^27 = 5$ (in $\mathbb{F}_{83}$).

5.5 The NFS in practice

The choice of the polynomial $f(x)$ in the NFS which is used to define the number field is of great importance. By taking care of what polynomial is chosen, the probability of finding enough relations can be greatly improved. Assuming we use the base-m method described in this chapter, some experiments are done in this section to show the influence of the choice of m on the number of smooth elements that are found.
We recall that to find a polynomial f with a root m modulo p, an m of size $p^{\frac{1}{\delta}}$ is chosen, where δ denotes the degree of the number field, and p is written in base m as:

$$p = \sum_{i=0}^{\delta} a_i m^i$$

where all a_i are positive constants smaller than m, and $a_\delta = 1$. The polynomial $f(x) = x^\delta + a_{\delta-1}x^{\delta-1} + \cdots + a_0$ with $a_i \in \mathbb{Z}$ now clearly satisfies $f(m) \equiv 0 \mod p$. However, the coefficients seem to be sort of random: for a slightly different m, the a_i can vary wildly. To demonstrate this, for the

prime $p = 1000059407$ we computed the number of created relations with the NFS using slightly different $m < p^{1/\delta}$. We set the values of the smoothness bound B and the bound on the sieving elements C equal to the optimal values found in section (5.3), and take the degree of our number field $\delta = 3$. The result is plotted in figure 5.1, while the polynomials corresponding to each m are listed in table 5.1.

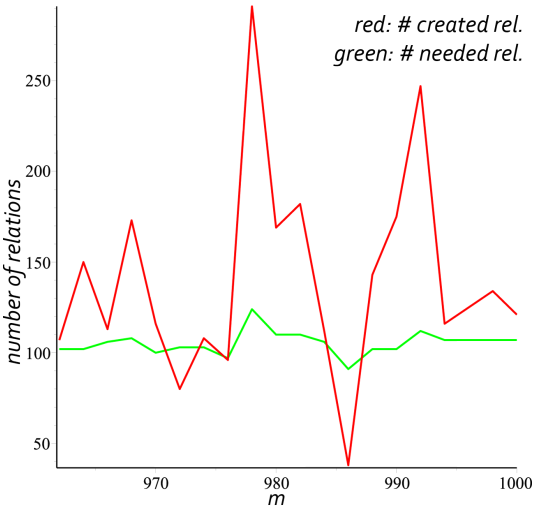


Figure 5.1: The number of created relations (red) and needed relations (green) for $f(x)$ created with the base-m method for varying m, $p = 1000059407$ and $\delta = 3$.

What is seen for this p is representative for what happens for all tested primes. The green line denotes the sum of $\pi(B)$ and $\mu(B)$ (number of first degree prime ideals of norm $\leq B$). Since B is fixed, $\pi(B)$ is fixed also so the variations in the line come from $\mu(B)$. With some exceptions, the red line is an exaggerated version of the green line: if the green line has a peek, then the red line seems more likely to have a peek also.
In general, rules of thumb for a good polynomial are: small (and balanced) coefficients, and high $\mu(B)$. If the coefficients of $f(x)$ are small, then this affects the value of $N(c + d\alpha) = (-d)^\delta f(-c/d)$ in a positive way. Indeed, the norm will be smaller giving it a higher probability of being smooth.
If there are many first degree prime ideals of norm $\leq B$, then the factor base consists of more elements, which again increases the probability of finding a smooth element.
The problem of finding a good polynomial is widely studied. Strategies to find a good polynomial can for example be found in [JLSV06] or [Kle06]. Usually, the strategies for finding good polynomials are the same for the NFS for integer factorization and the NFS for discrete logarithms.

m	$f(x)$	created	needed
962	$x^3 + 112x^2 + 142x + 23$	107	102
964	$x^3 + 118x^2 + 602x + 763$	150	102
966	$x^3 + 105x^2 + 672x + 179$	113	106
968	$x^3 + 99x^2 + 263x + 215$	173	108
970	$x^3 + 92x^2 + 849x + 77$	116	100
972	$x^3 + 86x^2 + 491x + 683$	80	103
974	$x^3 + 80x^2 + 159x + 37$	108	103
976	$x^3 + 73x^2 + 827x + 31$	96	97
978	$x^3 + 67x^2 + 545x + 617$	291	124
980	$x^3 + 61x^2 + 288x + 767$	169	110
982	$x^3 + 55x^2 + 56x + 427$	182	100
984	$x^3 + 48x^2 + 832x + 527$	112	106
986	$x^3 + 42x^2 + 651x + 33$	38	91
988	$x^3 + 36x^2 + 493x + 867$	143	102
990	$x^3 + 30x^2 + 361x + 17$	175	102
992	$x^3 + 24x^2 + 252x + 399$	247	112
994	$x^3 + 18x^2 + 167x + 977$	116	107
996	$x^3 + 12x^2 + 107x + 707$	125	107
998	$x^3 + 6x^2 + 71x + 533$	134	107
1000	$x^3 + 59x + 407$	121	107

Table 5.1: The polynomial $f(x)$ corresponding to the base-m expression for each m together with the number of created and needed relations for each polynomial, for $p = 1000059407$ and $\delta = 3$.

Chapter 6

The function field sieve

The function field sieve (FFS) is the function field analog of the number field sieve. In order to find discrete logarithms in finite fields $\mathbb{F}_{p^n}$, this algortihm makes use of an algebraic extension of $\mathbb{F}_p(X)$ in the same way that the number field sieve makes use of an algebraic extension of $\mathbb{Q}$. The NFS works well for finite fields $\mathbb{F}_{p^n}$ where $n \ll \log^{1/2} p$, while the FFS works well for $\mathbb{F}_{p^n}$ where $\log p \ll n^{1/2}$. We will see that in this case it achieves a complexity of $L_{p^n}[1/3, (32/9)^{1/3}]$.
First, function fields will be introduced in section 6.1 before the algorithm is presented in sections 6.2 and 6.3. After the running time analysis in section 6.4, the individual logarithm step is discussed in section 6.5 and an example of the FFS is given in 6.6. A recent adaption of the FFS that lead to the new index calculus algorithm as discussed in chapter 7 is called *pinpointing* and is discussed in 6.7. The last section, section 6.8, shows some results on how the FFS works in practice.

6.1 Function fields

An introduction to function fields can be found in [Sti09]. Here, we recall the definition:

Definition 6.1. An algebraic function field F/K of one variable over K is an extension field $F \supseteq K$ such that F is a finite algebraic extension of $K(X)$ for some element $X \in F$ which is transcendental over K.

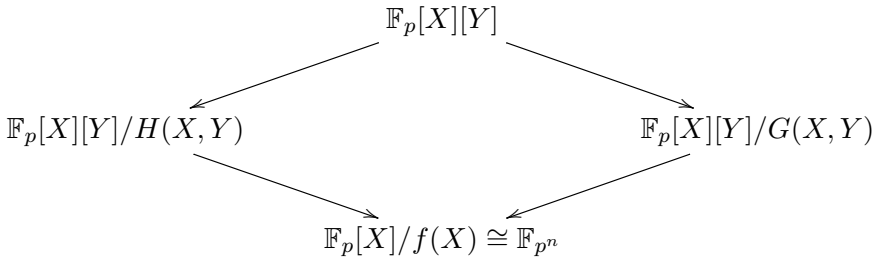
The simplest example of an algebraic function field is the *rational function field*. A function field F/K is called rational if $F = K(X)$ for some $X \in F$ which is transcendental over K. Each non-zero element $z \in K(X)$ has a unique representation

$$z = a \prod_i p_i(X)^{n_i}$$

in which $0 \neq a \in K$, the polynomials $p_i(X) \in K[X]$ are monic, pairwise distinct and irreducible, and $n_i \in \mathbb{Z}$.
Function fields can be represented by a simple algebraic field extension of a rational function field $K(X)$, i.e. $F = K(X, Y)$ where $\phi(Y) = 0$ for some irreducible polynomial $\phi(T) \in K(X)[T]$. If F/K is not a rational function field, it is not clear whether every element $0 \neq z \in F$ admits a decomposition into irreducibles as in the rational case. In fact, it is not even clear what we mean by an irreducible element. To formulate this problem for arbitrary function fields, (discrete) valuation rings are introduced.
In the function field sieve, two function fields are used. There is a so-called linear side, where we work with a rational function field, so seeking irreducible elements yields no problem here. On the other side, the so-called algebraic side, discrete valuation rings may be needed. The details of this are discussed in [Sch08] and [AH99]. In practice, such constructions will often be avoided, for example by taking both function fields rational. This will be shown in the example of section 6.6.

6.2 General idea

Assume that we want the logarithm of an element b in $\mathbb{F}_q$, where $q = p^n$ for some prime p. While using the function field sieve, elements of $\mathbb{F}_q$ are simply thought of as being polynomials. In this way, we can apply a notion of smoothness: an element of $\mathbb{F}_q$ is called B-smooth if it factors into irreducible elements which are of degree at most B, where B is some fixed smoothness bound. For simplicity, assume for now that we want to compute $\log_a b$, where $a, b \in \mathbb{F}_{p^n}^*$ are both B-smooth elements and $b \in \langle a \rangle$. By this restriction on b that it is B-smooth, we avoid the individual logarithm step (we will come back to this in section 6.5).
First, we choose a model for the finite field $\mathbb{F}_q$. This is done by fixing an irreducible polynomial $f \in \mathbb{F}_p[X]$ of degree n. Now, $\mathbb{F}_q$ can be represented by the set of polynomials in $\mathbb{F}_p[X]$ of degree $< n$, with addition and multiplication taken modulo f. The idea of the FFS is the following: Let the base field be denoted by $\mathbb{F}_p$, and let H and G be two monic irreducible polynomials in $\mathbb{F}_p[X][Y]$ such that their resultant in Y contains an irreducible factor $f(X)$ of degree n. Now, consider elements of the form $r(X) + s(X)Y \in \mathbb{F}_p[X][Y]$, where $r(X), s(X) \in \mathbb{F}_p[X]$ of degree less than a fixed bound. The relation collection step is based on the following commutative diagram, whose maps are ring homomorphisms:



Following this diagram, starting with an element of the form $r(X)+s(X)Y$ in $\mathbb{F}_p[X][Y]$, on the left-hand side we first obtain an element of $\mathbb{F}_p[X][Y]/H(X,Y)$. $\mathbb{F}_p[X][Y]/H(X,Y)$ is a Dedekind domain if and only if the affine curve C^{aff}, consisting of the zeros of $H(X,Y)$, is smooth. This is the case, and as $\mathbb{F}_p[X][Y]/H(X,Y)$ is a Dedekind domain, there is a unique factorization into prime ideals. The same happens if we follow the path on the right of the diagram, using G instead of H. Hence, we have two factorizations of the same element $r(X) + s(X)Y$.
Next, these two factorizations are pushed into the finite field $\mathbb{F}_p[X]/f(X) \cong \mathbb{F}_{p^n}$. This yields an equality between two products of elements of the finite field. If all these elements belong to the factor base, we have found a relation. The factor base consists of all prime ideals whose norms are of degree less that or equal to the smoothness bound. Then, when we've found enough of these multiplicative relations, they can be transformed into linear equations similar as with the NFS and they can be solved so that the logarithms of elements in the factor base are known.

6.3 The algorithm

Now, we follow the setup for the FFS as described in [Sch08] and [AH99].

Setup: constructing the function fields. First, let $f = X^n + g \in \mathbb{F}_p[X]$ where $g(X)$ is a polynomial of minimal degree (less than $n^{2/3}$) such that $X^n + g$ is irreducible and at least one root of g in some algebraic closure of $\mathbb{F}_p$ is of multiplicity one[1]. This f defines $\mathbb{F}_q \cong \mathbb{F}_p[X]/f(X)$. Now, let $F = \mathbb{F}_p(X)$, and build an extension of F by adjoining to it a root of a polynomial of degree d, where d is given by:

$$d = \lceil c_1^{-1} n^{1/3} \log^{-1/3} n \log^{1/3} p \rceil,$$

where c_1 will be determined in the complexity analysis section and we assume p does not divide d. We want an irreducible polynomial $H(X,Y) \in \mathbb{F}_p[X,Y]$

[1]It is simply assumed that such an f exists, and in practice they are easy to find (see also section 6.8).

of degree d in Y. This is done by taking a monic $m \in \mathbb{F}_p[X]$ of degree k, and writing $X^\delta f$ in base m for some $\delta > 0$. So, we try to write $X^\delta f$ as an expression in m:

$$X^\delta f = m^d + a_{d-1}m^{d-1} + \cdots + a_0,$$

where each $a_i \in \mathbb{F}_p[X]$ is of degree less than k. To create H, replace in the above expression m by Y, such that $H(X,Y) = Y^d + a_{d-1}(X)Y^{d-1} + \cdots + a_0(X)$. Now clearly $H(X,m) \equiv 0 \mod f$.
In our case, since f was chosen in such a special way, a suitable polynomial with particularly small coefficients can be found to build this extension in the following way: let k be the smallest multiple of d greater than or equal to n, so $k = \lceil \frac{n}{d} \rceil d$, hence $k-n = \delta$ for some $0 \leq \delta < d$. Then set $m = X^{\lceil \frac{n}{d} \rceil} = X^{k/d}$ and we see that $X^\delta f$ in base m becomes:

$$X^\delta f = m^d + X^{k-n}g.$$

Replacing m by Y gives $H = Y^d + X^{k-n}g(X) \in \mathbb{F}_p[X,Y]$. Due to the special demand on g that it has a root of multiplicity 1, by Eisenstein's criterion it follows that H is absolutely irreducible. Note that due to the construction, $H(X,m) \equiv 0 \mod f$. This sets up a ring homomorphism ϕ from the ring $\mathbb{F}_p[X,Y]/H(X,Y)$ to $\mathbb{F}_p[X]/f(X)$, sending Y to m. Note that we are (just as with the NFS) actually working with two polynomials which have a common root modulo f: indeed, if we let $G(X,Y) = Y - X^{k/d}$ then $G(X,m) \equiv 0 \mod f$ as well. This G is said to define the *linear side* of the sieve, H defines the *algebraic side*.
A slightly different way of constructing ϕ is the following: let $F = \mathbb{F}_p(X)$. Now, let $\bar{F}$ be an algebraic closure of F and let $\alpha \in \bar{F}$ be a root of H, considered as a polynomial in Y over F. Let $R = \mathbb{F}_p[X][\alpha]$ and denote by K the field of fractions of R. Then $\phi : R \to \mathbb{F}_q$ is the map which sends an element $h(X,\alpha)$ to the polynomial of degree $< n$ congruent to $h(X, X^{k/d})$ mod f. Since f divides $(X^{k/d})^d + X^{k-n}g$, we see ϕ is a ring homomorphism.

Sieving. Set the smoothness bound B at about $n^{1/3}$ (the exact value is determined in the complexity analysis section). The factor base S consists of all irreducible polynomials in $\mathbb{F}_p[X]$ of degree less than or equal to B. Take two elements $r, s \in \mathbb{F}_p[X]$ which are coprime and of degree bounded by

$$\deg(r), \deg(s) \leq c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p$$

(again, c_2 is determined later). We call a pair (r,s) B-smooth if both $r + sm$ and the norm of $r + s\alpha$ are B-smooth. This norm of $r + s\alpha$ is given by $s^d H(X, -r/s)$. We use a lattice sieve to check for smooth elements (for details, see [Pol93] or [JL02]). Note that testing a candidate for smoothness simply means factoring polynomials, which can be done in polynomial time

using the Berlekamp algorithm ([Ber70]), hence this step does not influence the overall complexity of the algorithm.
Now, all what's left to do before the linear algebra step is to compute valuation vectors. To define these, let M_F be the set of discrete valuations of $F = \mathbb{F}_p(X)$ of degree $\leq B$. Let M_K be the set of discrete valuations of K (the field of fractions of $\mathbb{F}_p[X][\alpha]$), which are extensions of the valuations in M_F. For each $(r, s) \in S$, we construct a vector $V_{r,s}$ containing the values $v(r + sm)$ for all $v \in M_F$, and the values $v(r + s\alpha)$ for all $v \in M_K$ (for details on the computation of these vectors, see [Sch08]).

Linear algebra. Recall that we are computing $\log_a b$, where b is B-smooth. Let V_a be the vector of length $|M_F| + |M_K|$ whose first $|M_F|$ entries are the values of $v(a)$ for $v \in M_F$ and whose remaining coordinates are 0. Define V_b in the same way but replace a by b. Let A be the matrix whose first column is V_a and remaining columns are the vectors $V_{r,s}$, and solve the congruence:

$$AX \equiv -V_b \mod (p^n - 1)/(p - 1).$$

Assuming that this yields a solution $(x, \dots, x_{r,s}, \dots)_{(r,s)\in S}$, we have that $x \equiv -\log_a b \mod (p^n - 1)/(p - 1)$. In order to see this, first let h be the gcd of $(p^n - 1)/(p - 1)$ and the class number of K, and assume $h = 1$ (else, use the modification as presented in [Sch02]). Let

$$\delta = a^x b \prod (r + sm)^{x_{r,s}}, \qquad \gamma = \prod (r + s\alpha)^{x_{r,s}},$$

where a, b are here considered as elements in $\mathbb{F}_p[X]$. Since

$$v(\delta) \equiv 0 \mod (p^n - 1)/(p - 1) \qquad \text{for all } v \in M_F,$$

we know δ is a product of an element in $\mathbb{F}_p^*$ and a $(p^n - 1)/(p - 1)$-st power. But any such power in $\mathbb{F}_q^*$ is in $\mathbb{F}_p^*$, hence $\phi(\delta) \in \mathbb{F}_p^*$. Since $h = 1$ and

$$v(\gamma) \equiv 0 \mod (p^n - 1)/(p - 1) \text{ for all } v \in M_K,$$

also γ is a product of an element in $\mathbb{F}_p^*$ and a $(p^n - 1)/(p - 1)$-st power, and $\phi(\gamma) \in \mathbb{F}_p^*$ as well. Now we see $a^x b = \phi(\delta)\phi(\gamma)^{-1} = \mu$ for some $\mu \in \mathbb{F}_p^*$, and indeed we have:

$$x \equiv -\log_a b \mod (p^n - 1)/(p - 1),$$

and all that remains is the computation of a logarithm in $\mathbb{F}_p^*$, namely $\log_{a'} \mu$, with $a' = a^{(p^n-1)/(p-1)}$. This is now a neglegible task.

6.4 Complexity analysis

The main idea for the analysis is to first fix an approximate size for the key parameters in the problem, such as the maximal degree of the elements $r(X)$

and $s(X)$, which depend on a constant that needs to be minimized. Then, determine the maximal size of the elements you're sieving over. Determine a lower bound for the probability P of elements of this size of being smooth (or, equivalently: the probability of finding a relation) and determine how many relations you need (i.e. the size of the factor base $\#S$). Then, assuming that you need $\#S/P$ trials to find enough relations, you need to check that there are more than $\#S/P$ pairs of (r, s) available (otherwise, the algorithm is expected to fail). With this, you should be able to find the minimum values for the unknown constants, and hence you have chosen your parameters such that the running time is optimized. You can now find an expression for the complexity of the sieving and linear algebra phase. The overall complexity is the maximum of the complexities of the sieving and linear algebra.
Note that such an analysis is a heuristic, asymptotic analysis. This means that it tries to predict a bound for the running time as p and n grow large, but due to several assumptions that have to be made it is not an exact analysis. Below, all steps where such assumptions are made are emphasized.

Maximal degree of sieving elements. Recall that we make a degree d extension of $\mathbb{F}_p[X]$, where d is given by:

$$d = \lceil c_1^{-1} n^{1/3} \log^{-1/3} n \log^{1/3} p \rceil.$$

Also, m is a polynomial of degree $\lceil n/d \rceil$, and the sieving elements r, s have a degree bounded by

$$c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p.$$

Now, we can give bounds on the degree of $r + sm$ and $r + s\alpha$. First, we consider $r + sm$. For the size of $\deg(m)$ we see:

$$\frac{n}{d} = \frac{n}{c_1^{-1} n^{1/3} \log^{-1/3} n \log^{1/3} p} = c_1 n^{2/3} \log^{1/3} n \log^{-1/3} p.$$

Clearly, the degree of $r + sm$ is dominated by the degree of sm. Note that $\deg(s)$ is small compared to $\deg(m)$, because:

$$n^{1/3} \log^{2/3} n \log^{-1/3} p < n^{2/3} \log^{1/3} n \log^{-2/3} p$$

(as $\lim_{n\to\infty}(\frac{n}{\log n})^{1/3} = \infty$). So, in total, $\deg(r + sm)$ is bounded by:

$$(c_1 + o(1)) n^{2/3} \log^{1/3} n \log^{-1/3} p.$$

Now, look at the norm of $r + s\alpha$. Recall that H was of the form $Y^d + X^{k-n} g(X)$, where $\deg(g(X)) < n^{2/3}$. The norm is given by:

$$N(r + s\alpha) = s^d H(X, -r/s) = (-r)^d + s^d X^{k-n} q(X).$$

Note that the degree is dominated by the degree of either r^d or s^d, which is bounded by:

$$\begin{aligned} d \cdot \deg(r) &= (c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p)(c_1^{-1} n^{1/3} \log^{-1/3} n \log^{1/3} p) \\ &= (c_2/c_1) n^{2/3} \log^{1/3} n \log^{-1/3} p. \end{aligned}$$

So, the total degree of $N(r + s\alpha)$ is bounded by:

$$(c_2/c_1 + o(1)) n^{2/3} \log^{1/3} n \log^{-1/3} p.$$

The total degree of a polynomial $(r + sm)N(r + s\alpha)$ is hence bounded by:

$$D = (c_2/c_1 + c_1 + o(1)) n^{2/3} \log^{1/3} n \log^{-1/3} p.$$

Now, let $Q = p^D$, and let the smoothness bound depend on Q: set $B = \log_p(L_Q[1/2, 1/\sqrt{2}])$.

Probability for smoothness. We adopt the heuristic assumption that with randomly chosen r and s, the polynomial $(r+sm)N(r+s\alpha)$ is random with degree bounded by D. The probability that a pair r, s is B-smooth (over S), is at least $L_Q[1/2; -1/\sqrt{2} + o(1)]$. This can be seen as follows. Let $N_q(n, m)$ denote the number of monic polynomials in $\mathbb{F}_q[X]$ of degree n and with each irreducible factor of degree at most m. The following theorem is copied from [BP98]:

Theorem 6.2. *Let $u = n/m$, and assume $1 \leq m < n$. Uniformly for all prime powers $q \geq (n \log^2 n)^{1/m}$, we have:*

$$N_q(n, m) = q^n / u^{(1+o(1))u}$$

as $m \to \infty$ and $u \to \infty$.

We see that the expected time to find one m-smooth polynomial of degree n is at most $u^{(1+o(1))u}$, or equivalently, a lower bound for the probability of finding an m-smooth polynomial is $u^{-(1+o(1))u}$. In our case, m equals the smoothness bound B. To find a lower bound for the probability we can only consider $n = D$, the maximal degree of $(r + sm)N(r + s\alpha)$. Now we try to find an expression for $u \log u = \frac{D}{B} \log \frac{D}{B}$. Note that (we can ignore $\log \log p$- and $\log \log D$-terms):

$$B = \log_p(L_Q[1/2, 1/\sqrt{2}]) \approx \frac{1}{\sqrt{2}} D^{1/2} \log^{1/2} D \log^{-1/2} p.$$

Then $D/B \approx \sqrt{2} D^{1/2} \log^{-1/2} D \log^{1/2} p$, so $\log D/B \approx 1/2 \log D$. This gives in total:

$$\begin{aligned} u \log u &= \frac{D}{B} \log \frac{D}{B} \\ &\approx \frac{\sqrt{2}}{2} D^{1/2} \log^{1/2} D \log^{1/2} p \\ &\leq \frac{1}{\sqrt{2}} \log^{1/2} p^D \log^{1/2} \log p^D, \end{aligned}$$

and indeed we have $u^{-(1+o(1))u} = L_Q[1/2; -1/\sqrt{2} + o(1)]$.

Obtaining enough relations. The number of elements in S and the number of discrete valuations over S are bounded by $L_Q[1/2; 1/\sqrt{2}]$. Hence, the number of (r, s)-pairs that needs to be tried follows from:

$$L_Q[1/2; 1/\sqrt{2}] \cdot 1/L_Q[1/2; -1/\sqrt{2} + o(1)] = L_Q[1/2; \sqrt{2} + o(1)].$$

Now, as the degree of r and s was bounded by $c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p$ and their coefficients are in $\mathbb{F}_p$, there are $p^{2c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p}$ pairs available. This leads to a constraint (as the number of pairs that needs to be tried cannot exceed the number of pairs available):

$$L_Q[1/2; \sqrt{2} + o(1)] \leq p^{2c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p}.$$

This can be rewritten (recall $Q = p^D$):

$$\begin{aligned} e^{\sqrt{2} \log^{1/2} p^D \log^{1/2} \log p^D} &\leq p^{2c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p} \\ 2D \log p \log \log p^D &\leq 4c_2^2 n^{2/3} \log^{4/3} n \log^{2/3} p \\ \frac{1}{2}(\frac{c_1^2 + c_2}{c_1}) \log \log p^D &\leq c_2^2 \log n. \end{aligned}$$

Looking at the $\log \log p^D$-term, we see:

$$\begin{aligned} \log \log p^D &= \log(D \log p) \\ &= \log(\frac{c_1^2 + c_2}{c_1}) + \frac{2}{3} \log n + \frac{1}{3} \log \log n + \frac{2}{3} \log \log p \\ &\leq \frac{2}{3} \log n. \end{aligned}$$

Filling this in then leads to:

$$\frac{c_1^2 + c_2}{3c_1} \leq c_2^2.$$

Solving this gives the minimal solution $c_1 = (2/3)^{1/3}$ and $c_2 = (4/9)^{1/3}$. Now, D can be determined as:

$$D = (2(2/3)^{1/3} + o(1)) n^{2/3} \log^{1/3} n \log^{-1/3} p.$$

Recall that $B = \log_p(L_Q[1/2, 1/\sqrt{2}])$, so B is bounded by:

$$B \leq \log_p L_{p^n}[1/3, (4/9)^{1/3} + o(1)],$$

Now that c_2 is determined, we can determine the complexity of the sieving phase. Recall that we sieve over $p^{2c_2 n^{1/3} \log^{2/3} n \log^{-2/3} p}$ elements, which can be written as:

$$\exp((2c_2 + o(1)) n^{1/3} \log^{2/3} n \log^{1/3} p),$$

which is bounded by:

$$L_{p^n}[1/3; (32/9)^{1/3} + o(1)].$$

Solving the linear system. The found relations form a sparse linear system which is solved modulo $(p^n - 1)/(p - 1)$. Then, all the logarithms of elements in the factor base are known up to a constant in $\mathbb{F}_p$. As this method assumed that p is small (compared to n), finding the logarithm in $\mathbb{F}_{p^n}^*$ is a negligible task.
Let $N = L_Q[1/2, 1/\sqrt{2}]$ be the number of columns of the matrix A. The linear system needs to be solved in time $O(N^2)$ to achieve an overall complexity of $L_{p^n}[1/3; (32/9)^{1/3} + o(1)]$:

$$\begin{aligned} N^2 &= L_Q[1/2, \sqrt{2}] \\ &= \exp(\sqrt{2}\log^{1/2} p^D \log^{1/2} \log p^D) \\ &\approx \exp(\sqrt{2} D^{1/2} \log^{1/2} p \log^{1/2} D) \\ &\approx \exp(\sqrt{2 \cdot 2(2/3)^{1/3} \cdot (2/3)}) n^{1/3} \log^{1/3} p \log^{2/3} n \\ &\approx L_{p^n}[1/3; (32/9)^{1/3}]. \end{aligned}$$

Note that if the system had to be solved modulo a prime number, it could be solved in time $O(N^2)$ ([Sch08]), but here it must be solved modulo $(p^n - 1)/(p - 1)$. As suggested in e.g. [JL02], the system could be solved modulo each large prime factor l of $(p^n - 1)/(p - 1)$ which can hence be done in time $O(N^2)$, leading indeed to a total complexity of $O(N^2)$ for the linear algebra phase.
So, solving the linear system with sparse matrix methods yields a running time which is asymptotically the same as the time of the sieving. Hence, the overall complexity of the algorithm is:

$$L_{p^n}[1/3; (32/9)^{1/3} + o(1)].$$

6.5 Individual logarithm step

Before, we assumed we wanted the logarithm of b in base a, where a and b were both B-smooth. Suppose now that b is not B-smooth. Then the linear system can be solved in such a way that the logarithm of all elements in the factor base is known. The logarithm of b can now be determined by a method called *special-q descent*. The idea is the following: start by building elements of the form $b^i t(X)^j$, where $t(X)$ is an element of the factor base and $i, j \in \mathbb{Z}_{>0}$. Note that such an element can be represented by rational fractions, whose numerators and denominators have degrees near $n/2$. Once such a representation is found, we test whether it can be factored in polynomials of smaller degree, say $\mu\sqrt{n}$. The optimal value for μ can be determined with

a complexity analysis[2], see [JL06], and is about $1/2 < \mu < 1$.
We proceed when we have found i and j such that $b^i Y^j$ factors like this. Then, let q be such a low degree polynomial in the factorization of $b^i Y^j$. We now find the logarithm of q by sieving again on elements of the form $r(X) + s(X)Y$, where $r(X), s(X)$ are polynomials of degree at most $\mu\sqrt{n}$ and chosen in such a way that q divides the linear side of the sieve. We want to find an element $r(X)+s(X)Y$ that factors (in both function fields) into polynomials of smaller degree. Then, for each element in this factorization with a degree still greater than B, we repeat this process until all degrees are lowered to below the smoothness bound. Then, we end up with a factorization into elements of the factor base, of which the logarithm is known. So, once the descent of all elements in the factorization of $b^i Y^j$ reaches a degree $\leq B$, we backtrack our steps and find the logarithm of our element b.

6.6 Example

The example given here follows [JL06]. It computes discrete logarithms in $\mathbb{F}_{q^n}$ where q is of the form p^k with p a medium-sized prime. There are two small differences with the method described above: the polynomial defining the algebraic side of the sieve is chosen differently and the sieving space contains elements of very small degree, which turns out to be enough in this case to find enough relations. The elements are only of the form $r(X) = wX + 1$, $s(X) = uX + v$, with $w, u, v \in \mathbb{F}_q$. This restriction on $r(X)$ comes from the fact that, since we are working with polynomials, all factorizations are defined up to a constant in the base field. By setting $r(X) = wX + 1$ we avoid multiple sieving over the same objects.

We will try the FFS in the field $\mathbb{F}_{41^9}$. As $9 = 3^2$, we seek two monic third-degree polynomials $H(X,Y), G(X,Y)$ such that $R_X = \mathrm{Res}_Y(H,G)$ has a monic irreducible factor of degree 9. By randomly trying we find:

$$\begin{aligned} G &= Y - g_1(X) = Y - X^3 - 40X^2 - 25X - 32 \\ H &= X + g_2(Y) = X + Y^3 + 2Y^2 + 4Y + 30 \\ R_X &= X^9 + 38X^8 + 37X^7 + 29X^6 + 32X^5 + 7X^4 + 30X^3 + 31X^2 + 26X + 1 \end{aligned}$$

Not that R_X is a monic irreducible factor of degree 9. Now $\mathbb{F}_{41^9} \cong \mathbb{F}_{41}[X]/R_X$, so this defines the extension field. Note that $Y = g_1(X)$ is a common root of G and H. Let $\alpha \equiv X \mod (R_X)$, so $R_X(\alpha) = 0$. Also, set $\beta = g_2(\alpha) = \alpha^3 + 2\alpha^2 + 4\alpha + 30$.
Note that G is linear in Y and H is linear in X, which means that $\mathbb{F}_p[X,Y]/G \cong$

[2]This must be optimized in order for the individual logarithm step not to become the bottleneck of the algorithm, so the complexity of this step should be less than the complexity of the sieving and linear algebra.

$\mathbb{F}_p[X]$ as we can write $Y = X^3+40X^2+25X+32$, and similarly $\mathbb{F}_p[X,Y]/H \cong \mathbb{F}_p[Y]$. Then the two function fields we are dealing with are rational function fields, so we do not have to worry about discrete valuations.
The smoothness basis contains on the linear side the 41 polynomials of the form $X+u$ with $u \in \mathbb{F}_{41}$. On the other side, due to our particular choice of the algebraic side, the smoothness basis also contains 41 elements, which are ideals of norm $X + g_2(u)$ with $u \in \mathbb{F}_{41}$. We'll seek multiplicative equalities between linear terms $(\alpha + a_i)$ and $(\beta + b_i)$, with $a, b, a_i, b_i \in \mathbb{F}_{41}$ as follows:

$$a\prod(\alpha - a_i) = b\prod(\beta - b_j). \tag{6.1}$$

This is an equality in $\mathbb{F}^*_{41^9}$. We take a generator g of $\mathbb{F}^*_{41^9}$. Note that $\mathrm{ord}(g) = q^n - 1$. In the above expression, we know $a \in \mathbb{F}^*_q$ so $\mathrm{ord}(a)|q-1$. This implies:

$$a = g^{\frac{q^n-1}{q-1}t},$$

where t is an unknown integer. In short, say $a = g^e$. Also, all linear terms $(\alpha - a_i)$ for $a_i \in \mathbb{F}_{41}$ can be written in terms of the generator, so $(\alpha - a_i) = g^{e_i}$. Likewise for $b = g^f$, $(\beta - b_i) = g^{f_i}$ for $b_i \in \mathbb{F}_{41}$.
Now, the multiplicative equalities can be transformed into linear ones, using the facts that $e, f \equiv 0 \mod \frac{q^n-1}{q-1}$ and $e_i, f_i \in \mathbb{Z}/(\mathrm{ord}(g))\mathbb{Z}$. Now equation (6.1) transforms into:

$$g^{e+\sum e_i} = g^{f+\sum f_i},$$

which gives rise to the desired linear equality:

$$\sum e_i \equiv \sum f_i \mod (q^n-1)/(q-1).$$

When we have enough equations of the form (6.1), we can continue with the linear algebra phase. After that, we'll have an individual logarithm step.

Relation collection. To collect relations, we take as our generator $g = \alpha$ and try elements of the form $XY + aX + bY + c$, where $a, b, c \in \mathbb{F}_{41}$. For such an element, replace every Y for $g_1(\alpha)$, so the resulting equation is a 4^{th} degree polynomial in α. Likewise, replace every X by $g_2(\beta)$. Whenever both norms are smooth, we can write an explicit identity between generators. For example, this gives ($a = 1, b = 27, c = 30$):

$$(\alpha+30)(\alpha+10)(\alpha+16)(\alpha+11) = (\beta+39)(\beta+22)(\beta+12)^2.$$

We have a relation between four linear terms $(\alpha+a)$ and four terms $(\beta+b)$. In total, there are 177 good relations, while we only need 81 linear independent relations.

Linear algebra. We put the exponents of all relations in a matrix: in the first 41 columns the exponents of the $(\alpha + a)$-terms, in the last 41 columns *minus* the exponents for the $(\beta + b)$-terms, as:

$$(\alpha+30)(\alpha+10)(\alpha+16)(\alpha+11)(\beta+39)^{-1}(\beta+22)^{-1}(\beta+12)^{-2} = 1,$$

which is an equality in $\mathbb{F}_{q^n}^*$. The matrix has rank $2q-1$ over the rationals, as there is a parasitic solution due to the special form of equations we use. Indeed, the sum of the first 41 elements in each row is 4, and of the last 41 elements is -4. So one solution is simply 4 for every unknown. Therefore, we also enter a relation with a different structure. The easiest way to do this is to look for a linear polynomial which completely splits into the function field defined by H, for example, we can add the asymmetric relation:

$$(\alpha + 17) = (\beta + 8)^2(\beta + 27).$$

Now, the matrix has full rank. We take as a generator $g = \alpha$, so we can add an extra column (make an augmented matrix) and add the known solution $\alpha = g^1$:

$$\begin{pmatrix} 1 & 0 & \dots & 0 & | & 1 \end{pmatrix},$$

(with 81 zeros) to make the system solvable.
Putting the resulting matrix in reduced row echelon form modulo $\frac{q^n-1}{q-1}$ gives the power of the generator for each linear term up to a multiplicative constant (so modulo $\frac{q^n-1}{q-1}$). For example:

$$\log_g(\alpha + 1) \equiv 7983947282245 \mod \frac{q^n-1}{q-1}$$

$$\log_g(\beta) \equiv 2212517000985 \mod \frac{q^n-1}{q-1}$$

Individual logarithm. We want to know the logarithm of:

$$b = X^5 + 3X^4 + 17X^3 + 37X + 9$$

This is irreducible modulo 41, so we use the special-q descent. The first step is to find elements of the form $z(X) = b(X)t(X)^i$ where $t(X)$ is an element of the factor base, and then write this $z(X)$ as a fraction such that the numerator and denominator have degree ≈ 3. However, as $n = 9$ in our case, it is easy to find a $z(X)$ which splits itself into polynomials of maximal degree 3, we skip this step. We see:

$$\begin{aligned} z(X) = b(X)X^7 &= 36X^8 + 29X^7 + 24X^6 + 32X^5 + 7X^4 + \\ &\quad 30X^3 + 18X^2 + X + 11 \\ &= 36(X^2 + 36X + 20)(X^3 + 25X^2 + 6X + 38) \\ &\quad (X + 17)(X + 22)(X + 9) \end{aligned}$$

(The polynomials are considered in $\mathbb{F}_{41}[X]/R_X$). As we want the answer modulo $\frac{q^n-1}{q-1}$, the constant 36 can be forgotten. Now, we start the descent on the non-linear terms in the factorization of $z(X)$, so on $(X^2+36X+20)$ and $(X^3+25X^2+6X+38)$. First, for $(X^2+36X+20)$, consider the element:

$$(\alpha+7)\beta-(32\alpha^2+10\alpha+9).$$

This is equal to (replacing β by $f(\alpha)$):

$$(\alpha+35)(\alpha+17)(\alpha^2+36\alpha+20).$$

And also to (replacing α by $f(\beta)$):

$$9(\beta+6)(\beta+32)(\beta+24)(\beta+36)(\beta+9)(\beta+20).$$

This way, we can compute the logarithm of $(X^2+36X+20) \mod \frac{q^n-1}{q-1}$, as all the other terms are linear terms. They are in the factor base, and their logarithms are known. Indeed,

$$\begin{aligned}\log(X^2+36X+20) &\equiv -\log(\alpha+35)-\log(\alpha+17)+\log(\beta+6)+\log(\beta+32)\\ &\quad +\log(\beta+24)+\log(\beta+36)+\log(\beta+9)+\log(\beta+20)\\ &\equiv 5239839337627 \mod \frac{q^n-1}{q-1}.\end{aligned}$$

We do the same for the other non-linear term of $z(X)$, namely $(X^3+25X^2+6X+38)$. First, consider the element:

$$(\alpha+21)\beta-(16\alpha+1).$$

This is equal to

$$(\alpha^3+25\alpha^2+6\alpha+38)(\alpha+36).$$

And also to:

$$(\beta+24)(\beta^2+27\beta+15)(\beta+17).$$

Hence, we've *lowered the degree* in the sense that we now only need an equation in which $\beta^2+27\beta+15$ splits into linear terms. So, one more step in our descent. Take the element:

$$(\alpha+16)\beta-(33\alpha^2+37\alpha+6).$$

This is equal to

$$(\alpha+13)^2(\alpha+18)(\alpha+12).$$

And also to:

$$8(\beta+11)(\beta+10)(\beta+19)^2(\beta^2+27\beta+15).$$

Thus completing the descent. The logarithm of $b(X)$ can now be found by retracing our steps, being:

$$\log_\alpha b \equiv 7405183113737 \mod \frac{q^n-1}{q-1}.$$

6.7 Adaption: pinpointing

Although the FFS has a good expected running time, it can be argued that sieving is not a very efficient method. Even if an element is not smooth, still some time is spent on it while marking it as a bad element. Hence, a different method for generating the relations was proposed in [Jou13]. This method, which is not a sieve anymore, works in finite fields of the form $\mathbb{F}_q$ with $q = p^n$ for a medium prime p. The idea is the following: we restrict the polynomials to have te following form:

$$\begin{aligned} X &= Y^{d_1} \\ Y &= g_2(X), \end{aligned}$$

where g_2 is a degree d_2-polynomial. To generate relations we consider the space spanned by XY, X, Y and 1, so we are (after renormalization) considering the candidates:

$$Y^{d_1+1} + aY^{d_1} + bY + c = Xg_2(X) + aX + bg_2(X) + c, \tag{6.2}$$

where a, b and c are arbitrary coefficients in $\mathbb{F}_q$. Now, if we perform a change of variable $Y = aU$, we see that the left-hand side splits into linear terms if and only if $U^{d_1+1} + U^{d_1} + ba^{-d_1}U + ca^{-d_1-1}$ factors into linear terms. So, search for a polynomial in U of the form $U^{d_1+1} + U^{d_1} + BU + C$ which is smooth. Once we've found one such smooth polynomial, we can amplify it into many smooth polynomials using the substitution $U = Y/a$, of the form:

$$Y^{d_1+1} + aY^{d_1} + bY + c,$$

where a is any nonzero element in $\mathbb{F}_q$, $b = Ba^{d_1}$ and $c = Ca^{d_1+1}$. So, finding one smooth polynomial leads directly to finding approximately $q/(d_2 + 1)!$ smooth polynomials.
Here, we considered only the left-hand side of equation (6.2). With some specific extension fields, also a similar tric can be used on the right-hand side of this equation, speeding up the relation collection phase even more. For details, see [Jou13]. This method, which is called *pinpointing*, lead to the new index calculus algorithm as introduced in the next chapter.

6.8 The FFS in practice

In this section some aspects of the FFS in practice are discussed. First, note that sometimes the theory cannot be applied as easily in practice as one would hope. When working with the FFS, discrete valuations are usually not built-in functions in software, so when implementing this it would be nice if they can be avoided. For example, the algorithm described by [JL06] avoids this by using two rational function fields.

Next, we check whether the constructions for the polynomials mentioned above indeed yield enough relations. We consider the algorithm described in section 6.3. Using the bounds as given in the complexity analysis section, we can check whether these are sufficient to find enough relations. First, note that an upper bound for the number of relations needed is given by $(d+1)p^B$, since there are approximately p^B irreducible polynomials of degree $< B$ and there lie at most d points above each point in $\mathbb{F}_p(X)$, where d is the degree of the extension (see e.g. [Sti09, $Cor.$ 3.1.12a]).
In table 6.1, the number of relations found for $p = 2$ but different n (using the smallest polynomial $f(X)$ that meets the demands) is listed. This varies wildly, and the particular choice of n seems to have a great impact on the number of relations. The degree of $g(X)$ seems of great influence: the smaller the degree of $g(X)$, the higher seems the probability of finding relations. Note, however, that n is fixed before we start the algorithm. So, in the unlucky case where only $g(X)$ of rather high degree can be found, we may have to increase some bounds, such as the smoothness bound or the maximal degree of $r(X)$ and $s(X)$ to find enough relations.

n	$H(X,Y) = Y^3 + g(X)$	# rel. found	# rel. needed
60	$Y^3 + X + 1$	1790	2048
61	$Y^3+X^5+X^4+X^3+X^2+X+1$	1158	2048
62	$Y^3+X^7+X^6+X^5+X^4+X+1$	643	2048
63	$Y^3 + X + 1$	3215	2048
64	$Y^3 + X^2(X^4 + X^3 + X^2 + 1)$	2078	2048
65	$Y^3 + X(X^8 + X^7 + X^3 + X^2 + X + 1)$	1900	4096
66	$Y^3 + X^3 + 1$	5715	4096
67	$Y^3 + X^2(X^7 + X^6 + X^5 + X^2 + X + 1)$	2701	4096

Table 6.1: Different n and the number of found relations for $p = 2$.

As a demonstration of the influence of the degree of $g(X)$, table 6.2 shows for $p = 2$ and $n = 63$ the smallest polynomials such that $f(X)$ meets the demands of section 6.3. In the notation of this section, recall that we sieve over polynomials $r(X)$, $s(X)$ such that $r + sm$ and $N(r + s\alpha) = s^d H(X, -r/s)$ are B-smooth. We see that when the degree of $g(X)$ is lower, it seems more likely that there are more relations found. This trend, that a polynomial with a low degree gives the most relations, is seen for all tested n for small primes, and is consistent with the fact that the probability of being smooth decreases as the size of the elements grows. Hence, taking $g(X)$ as small as possible increases the probability of finding relations.

$H(X,Y)$	# rel. found	# rel. needed
Y^3+X+1	3215	2048
Y^3+X^5+1	2055	2048
$Y^3+X^6+X^5+X^3+X^2+X+1$	2109	2048
$Y^3+X^9+X^3+X+1$	1594	2048
$Y^3+X^9+X^7+X^5+X^3+X+1$	1865	2048

Table 6.2: The influence of different $g(X)$ on the number of found relations for $p = 2$, $n = 63$.

Chapter 7

A new index calculus algorithm for small characteristic

Based on the idea of pinpointing for medium primes (see section 6.7), in [BGJT14] a new index calculus algorithm is presented for solving discrete logarithms in finite fields of small characteristic. The main idea is to choose a polynomial that splits by construction into linear terms, and use homographies to create relations out of this specific polynomial. Then, using a sparse representation of the finite field, high degree elements are transformed into low degree elements, increasing the probability that a relation between linear polynomials is found. This is explained in section 7.1.
In section 7.2, we will describe the (in 2014) asymptotically fastest method, as introduced in [BGJT14]. In section 7.3 it will be shown that in fields of the form $\mathbb{F}_{q^{2k}}$ where $q \approx k$, this is a quasi-polynomial algorithm. In practice, it might be more efficient to use an adaption of this method, such as in [Jou14] or [GKZ14]. The main advantages of those methods are described in section 7.4, together with some open questions regarding this new algorithm. An example of the algorithm is given in section 7.5.

7.1 General idea

The new algorithm computes discrete logarithms in a finite field of the form $\mathbb{F}_{q^{2k}}$, where q is a power p^l of a small prime p, and $q \approx k$. The two key ingredients of the new index calculus algorithm are a specific choice of a polynomial over $\mathbb{F}_q$ that splits by construction into linear terms, the so-called *systematic equation*, which generates many relations when it is transformed in a specific way, and an efficient degree k extension of $\mathbb{F}_{q^2}$ to represent $\mathbb{F}_{q^{2k}}$.

Generate relations. We choose a polynomial $f(X) \in \mathbb{F}_q[X]$ as our systematic equation which splits naturally into linear terms:

$$f(X) = X^q - X = \prod_{\alpha \in \mathbb{F}_q} (X - \alpha). \tag{7.1}$$

With pinpointing, a linear change of variables was used to transform a single good polynomial g that yielded a relation into several such polynomials, sending $g(X)$ to $g(aX)$ for any non-zero constant $a \in \mathbb{F}_q$. This can be done for a larger class of change of variables induced by homographies: consider the extension $\mathbb{F}_{q^2} \supset \mathbb{F}_q$ and send $X \mapsto \frac{aX+b}{cX+d}$ for $(a, b, c, d) \in \mathbb{F}_{q^2}^4$ satisfying $ad \neq bc$. The condition $ad \neq bc$ guarantees that the map $X \mapsto \frac{aX+b}{cX+d}$ defines an automorphism of $\mathbb{F}_{q^2}(X)$. Note, however, that this will not transform a polynomial g into a polynomial again. So instead we perform a homogeneous evaluation of g at $\frac{aX+b}{cX+d}$. We consider:

$$G_{abcd}(X) = (cX + d)^{\deg(g)+1} g\left(\frac{aX + b}{cX + d}\right) \in \mathbb{F}_{q^2}[X]. \tag{7.2}$$

Since we work with the polynomial (7.1), we take $(a, b, c, d) \in \mathbb{F}_{q^2}^4$ because when $(a, b, c, d) \in \mathbb{F}_q^4$ we get a multiple of $f(X)$ after the transformation. Indeed, then $a^q = a$ and similar for b, c and d, and $f(X)$ is then transformed into $F_{abcd}(X) = (ad - bc)(X^q - X)$.
Geometrically, the change of variables we consider here is equivalent to writing a polynomial equation for the image of the projective line $\mathbb{P}^1(\mathbb{F}_q)$ by an element of the projective linear group $\mathrm{PGL}_2(\mathbb{F}_{q^2})$. This way of looking at it is explored further in the next section, where the algorithm is described.

Define extension field. The degree k extension from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^{2k}}$ is made using a degree k irreducible factor $I(X)$ of $h_1(X)X^q - h_0(X)$, where $h_1(X)$ and $h_0(X)$ are coprime polynomials of small degree with coefficients in $\mathbb{F}_{q^2}$. If we let $\delta = \max(\deg h_0, \deg h_1)$, we see that $k \leq \delta + q$.[1] Note that in this construction we have the relation:

$$X^q \equiv \frac{h_0(X)}{h_1(X)} \mod I(X).$$

The image of $X^q - X$ by the transformation (7.2) is a polynomial which is of the form $a_{q+1}X^{q+1} + a_qX^q + a_1X + a_0$, for $a_i \in \mathbb{F}_{q^2}$, $i \in \{q+1, q, 1, 0\}$ (all other terms are zero). With the given finite field representation of $\mathbb{F}_{q^{2k}}$, this polynomial yields an element of $\mathbb{F}_{q^{2k}} \cong \mathbb{F}_{q^2}/(I(X))$ represented as a fraction of polynomials of low degrees in $\mathbb{F}_{q^2}[X]$. Indeed, when we transform the

[1] Note that it is a heuristic assumption that such an $I(X)$ for a small value of δ can be found. In practice, it turns out to be rather easy to find good h_0 and h_1 ([Jou14],[BGJT14]) and it has to be done only once during the set-up of the algorithm, so spending some time on finding good h_0 and h_1 does not change the asymptotic complexity of the algorithm.

systematic equation (7.1) using the transformation as defined in (7.2), we get:

$$\begin{gathered}(cX+d)\prod_{\alpha\in\mathbb{F}_q}((a-\alpha c)X+(b-\alpha d)= \\ (aX+b)^q(cX+d)-(aX+b)(cX+d)^q.\end{gathered} \tag{7.3}$$

The right-hand side can be evaluated to:

$$\begin{gathered}(ca^q-ac^q)X^{q+1}+(da^q-bc^q)X^q+(cb^q-ad^q)X+(db^q-bd^q)\equiv \\ \frac{(ca^q-ac^q)Xh_0(X)+(da^q-bc^q)h_0(X)+(cb^q-ad^q)Xh_1(X)+(db^q-bd^q)h_1(X)}{h_1(X)} \\ \bmod I(X).\end{gathered}$$

Hence, equation (7.3) yields an equality in $\mathbb{F}_{q^{2k}}$ between a product of linear polynomials over $\mathbb{F}_{q^2}$ and a fraction with low-degree numerator and denominator h_1, which is independent of $a, b, c, d \in \mathbb{F}_{q^2}$. In this way, relations can easily be found by searching for tuples (a, b, c, d) such that the numerator of the fraction splits. For example, in order to find the logarithms of linear terms (considering $h_1(X)$ as an element of our smoothness basis), we get a multiplicative relation when the numerator of the fraction also factors into linear terms.

7.2 The algorithm

To apply the ideas from the previous section, the finite field in which we want to compute the discrete logarithm needs to be of a special form. Indeed, it needs to have a large enough subfield and we need to be able to find a sparse representation for it. This is formalized in the following definition (from [BGJT14]).

Definition 7.1. A finite field K admits a *δ-sparse medium subfield representation* if:

- it has a subfield of q^2 elements for a prime power q, i.e. K is isomorphic to $\mathbb{F}_{q^{2k}}$ with $k \geq 1$;
- there exist two coprime polynomials $h_0(X)$ and $h_1(X)$ over $\mathbb{F}_{q^2}$ of degree $\leq \delta$ such that $h_1(X)X - h_0(X)$ has a degree k irreducible factor $I(X)$.

For now, assume that all finite fields we consider admit a δ-sparse medium subfield representation (in section 7.3 it is explained how any finite field can be embedded into a field that admits a δ-sparse medium subfield representation). Furthermore, the degrees of $h_0(X)$ and $h_1(X)$ are bounded by a

constant δ which is assumed to be independent[2] of q and k. In the fields we consider, elements are represented as polynomials of degree less than k with coefficients in $\mathbb{F}_{q^2}$.
For a finite field $K = \mathbb{F}_{q^{2k}}$ that admits a δ-sparse medium subfield representation, given any primitive root $g \in \mathbb{F}_{q^{2k}}$ (so we consider discrete logarithms in base g), there is a heuristic algorithm which is polynomial in q and k under two heuristics that are explained below and that can be used for the following two tasks:

1. The algorithm returns the logarithm of $h_1(X)$ and the logarithms of all the elements of $\mathbb{F}_{q^{2k}}$ of the form $X + a$, $a \in \mathbb{F}_{q^2}$.

2. Given an element of $\mathbb{F}_{q^{2k}}$ represented by a polynomial $P \in \mathbb{F}_{q^2}[X]$ with $2 \leq \deg P \leq k-1$, the algorithm returns an expression of $\log P(X)$ as a linear combination of at most $O(q^2k)$ logarithms $\log P_i(X)$ with $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$ and of $\log h_1(X)$.

This algorithm is a building block for a complete discrete logarithm algorithm in the following way: let $P(X)$ be the element of K of which we want to compute the discrete logarithm. After we've applied the algorithm once, we obtain a relation of the form:

$$\log P = e_0 \log h_1 + \sum e_i \log P_i,$$

where each $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$ and the sum has $O(q^2k)$ terms. Then we can apply the algorithm on the P_i's, hence creating a descent tree. At the end, the logarithm is expressed as a linear combination of logarithms of h_1 and linear polynomials, for which the logarithms are computed with the algorithm. By backtracking the steps through the descent tree, the discrete logarithm of $P(X)$ is found.
Now, we arrive at the description of the algorithm. Assume we want to compute the discrete logarithm of $P(X) \in \mathbb{F}_{q^2}[X]$ with $2 \leq \deg P \leq k-1$. The strategy is to find relations between $P(X)$ and its translates by a constant in $\mathbb{F}_{q^2}$. Recall that the systematic equation is:

$$X^q - X = \prod_{\alpha \in \mathbb{F}_q} (X - \alpha).$$

We view this equation as involving the projective line $\mathbb{P}^1(\mathbb{F}_q)$. Let $S = \{(\alpha, \beta)\}$ be a set of representatives of the $q+1$ points $(\alpha : \beta) \in \mathbb{P}^1(\mathbb{F}_q)$, chosen such that the following homogeneous equality holds:

$$X^q Y - X Y^q = \prod_{(\alpha,\beta) \in S} (\beta X - \alpha Y). \tag{7.4}$$

[2] In [BGJT14] heuristic arguments are provided that $\delta = 2$ will work for any finite field of the form $\mathbb{F}_{q^{2k}}$ with $k \leq q+2$, but in general, letting δ be a constant independent of q and k is sufficient for the complexity to hold.

To make translates of $P(X)$ appear, we consider the action of homographies. Homographies can be introduced in terms of a matrix $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Such an m acts on $P(X)$ as:

$$m \cdot P = \frac{aP + b}{cP + d}$$

(this is the action of m on elements of $\mathbb{P}^1(\mathbb{F}_{q^{2k}})$). Note that multiplying all entries of m by a non-zero constant does not change its action on $P(X)$. Since we consider equation (7.4) and we are going to replace $X = aP+b, Y = cP + d$, for the same reasons as described in section 7.1, m should not have entries that are all defined over $\mathbb{F}_q$ and m has to be invertible (i.e. $ad - bc \neq 0$). Therefore the matrices of the homographies that we consider are going to be taken in the cosets

$$P_q = \mathrm{PGL}_2(\mathbb{F}_{q^2})/\mathrm{PGL}_2(\mathbb{F}_q).$$

To each $m \in P_q$ we associate the following equation, obtained by substituting $X = aP + b$ and $Y = cP + d$ in equation (7.4). This now becomes:

$$\begin{aligned} (aP+b)^q(cP+b) - (aP+b)(cP+d)^q &= \prod_{(\alpha,\beta)\in S} (\beta(aP+b) - \alpha(cP+d)) \\ &= \prod_{(\alpha,\beta)\in S} ((-c\alpha + a\beta)P - (d\alpha - b\beta)) \\ &= \lambda \prod_{(\alpha,\beta)\in S} (P - x(m^{-1} \cdot (\alpha : \beta))). \end{aligned} \tag{7.5}$$

The last line needs some comment. Here, $\lambda \in \mathbb{F}_{q^2}$ is the constant which makes the leading terms of the expressions considered as polynomials in P of the two sides match. Then, $P - x(m^{-1} \cdot (\alpha : \beta))$ denotes $P - u$ when $m^{-1} \cdot (\alpha : \beta) = (u : 1)$ (whence we have $u = \frac{d\alpha - b\beta}{-c\alpha + a\beta}$), or 1 if $m^{-1} \cdot (\alpha : \beta) = \infty$. This may occur when $(a : c) \in \mathbb{P}^1(\mathbb{F}_q)$, since then $(-c\alpha + a\beta)$ vanishes for a point $(\alpha : \beta) \in \mathbb{P}^1(\mathbb{F}_q)$. In this case, one of the factors of the product contains no term in $P(X)$ but only a constant term. Hence, the right-hand side of equation (7.5) is, up to a multiplicative constant, a product of $q + 1$ or q translates of $P(X)$ by elements of $\mathbb{F}_{q^2}$.
The left-hand side of equation (7.5) can be written as a smaller degree equivalent using the special form of our defining polynomial. In $\mathbb{F}_{q^{2k}}$ we have $X^q \equiv \frac{h_0(X)}{h_1(X)} \mod I(X)$. Let us denote $\tilde{a} = a^q$ when a is any element of $\mathbb{F}_{q^2}$. Furthermore, we write $\tilde{P}(X)$ the polynomial $P(X)$ with all its coefficients raised to the power q. The left-hand side of equation (7.5) can now be written as:

$$(\tilde{a}\tilde{P}(X^q) + \tilde{b})(cP + b) - (aP + b)(\tilde{c}\tilde{P}(X^q) + \tilde{d}),$$

which is congruent modulo $I(X)$ to:

$$L_m := \left(\tilde{a}\tilde{P}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{b}\right)(cP+b) - (aP+b)^q \left(\tilde{c}\tilde{P}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{d}\right). \quad (7.6)$$

Clearly, the denominator of L_m is a power of h_1 and its numerator has degree at most $(1+\delta)\deg P$, where $\delta = \max(\deg h_0, \deg h_1)$.

1. Logarithms of linear elements. To achieve the first purpose of the algorithm, finding the logarithms of all elements of the form $X + a, a \in \mathbb{F}_{q^2}$ and of h_1, one simply sets $P(X) = X$ (note that the equations become essentially the same as (7.3)). We obtain a relation if the numerator of L_m splits into linear terms, then the equation contains on both sides only linear polynomials and h_1. They can be transformed into a linear system whose unknowns are $\log(X + a)$ with $a \in \mathbb{F}_{q^2}$ and $\log h_1$. We rely on the following heuristic to ensure that the linear system has a unique solution:

Heuristic 7.2. The linear system constructed from all the equations (7.5) for $P(X) = X$ has full rank.

Note that solving the linear system yields only the ratio between the discrete logarithms of the linear polynomials and of $h_1(X)$, as we are solving a homogeneous system of equations. Hence, we also need a inhomogeneous equation, which can be found using the given primitive root g of $\mathbb{F}_{q^{2k}}$. We clearly know the discrete logarithm of g in base g: $\log_g g = 1$. If g is also a linear polynomial or equals h_1, then all discrete logarithms are found immediately. Hence, in practice, such a primitive root g is preferred. Otherwise, if g is not a linear polynomial or h_1, then an extra relation between g and only linear polynomials and h_1 needs to be found to find the discrete logarithms.

2. Writing $\log P$ as a sum of lower degree elements. Now, we want to achieve the second purpose of the algorithm, i.e. express $\log P(X)$ as a linear combination of $\log h_1(X)$ and $\log P_i(X)$, with $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$. We say $m \in P_q$ yields a relation when the numerator of L_m is $\lceil \frac{1}{2} \deg P \rceil$-smooth. Take at most one matrix m that yields a relation in each coset of P_q. Construct a row vector $v(m)$ of dimension q^2+1 for each of those $m \in P_q$ in the following way. Index the coordinates by $\mu \in \mathbb{P}^1(\mathbb{F}_{q^2})$. Associate the value 1 to μ if $P - x(\mu)$ appears in the right-hand side of equation (7.5) (or, equivalently, if $\mu = m^{-1} \cdot (\alpha : \beta)$ with $(\alpha : \beta) \in \mathbb{P}^1(\mathbb{F}_q)$), otherwise the value 0. and then associate to a polynomial P a matrix $H(P)$ whose rows are the vectors $v(m)$ for each m. Then, we rely on the following heuristic.

Heuristic 7.3. For any $P(X)$, the set of rows $v(m)$ for cosets $m \in P_q$ that yield a relation form a matrix $H(P)$ which has full rank $q^2 + 1$.

Next we perform linear algebra on this matrix. Since $H(P)$ has full rank, the vector $(\dots, 0, 1, 0 \dots)$ with the 1 corresponding to $P(X)$ (or, equivalently, to the μ such that $x(\mu) = 0$) can be written as a linear combination of the rows. Each row can be replaced by the logarithm of the corresponding factorization of L_m. Then we have written $\log P(X)$ as a linear combination of $\log h_1(X)$ and $\log P_i$, where P_i are elements of degree $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$ appearing in the factorization of L_m.
Now, we show that there appear at most $O(q^2 k)$ of such P_i's in this equation. Since there are $O(q^2)$ columns, the elimination process involves at most $O(q^2)$ rows. Each row corresponds to an equation (7.5), so it involves at most $\deg L_m \leq (1+\delta) \deg P$ polynomials on the left-hand side. In total, the polynomial P is expressed as a linear combination of at most $O(q^2 \cdot \deg P) \leq O(q^2 k)$ polynomials.
Finally, we only need to look at the claim that this algorithm is polynomial in q and k. First, note that we've ignored the λ's appearing in equation (7.5) so far. We need to compute the discrete logarithms of the λ's, but these are elements of $\mathbb{F}_{q^2}^*$ and can certainly be computed in polynomial time in q. Furthermore, for the rest of the algorithm, note that the order of $\mathrm{PGL}_2(\mathbb{F}_{q^i})$ is $q^{3i} - q^i$, so the sets of cosets P_q has $\frac{q^6 - q^2}{q^3 - q} = q^3 + q$ elements. For each m, testing whether (7.5) yields a relation amounts to some polynomial manipulation and a smoothness test, which is done in polynomial time in q and the degree of $P(X)$, which is bounded by k. Finally, the linear algebra step can be done in $O(q^5)$ operations using sparse matrix techniques as we have $q + 1$ non-zero entries per row and a size of $q^2 + 1$. Hence, the overall complexity is polynomial in q and k.

7.3 Complexity analysis

The total algorithm to compute the discrete logarithm of an element $P(X) \in \mathbb{F}_{q^{2k}}$ consists of applying the algorithm described in the previous section multiple times. After the algorithm is applied once, we obtain a relation of the form:

$$\log P = e_0 \log h_1 + \sum e_i \log P_i,$$

where each $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$ and the sum has at most $O(q^2 k)$ terms. Then the algorithm is applied on each of the P_i's, creating a descent tree with an arity in $O(q^2 k)$. At the end, the logarithm is expressed as a linear combination of logarithms of h_1 and linear polynomials, for which the logarithms were computed with the algorithm. Note that at the last nodes of the descent tree there are only linear polynomials, while at the other nodes (the *internal nodes*) we still have higher degree polynomials so at those nodes we have to apply the algorithm.
The complexity of this total algorithm is given by the number of internal nodes times the complexity of the algorithm which is once applied at each

internal node, and which is polynomial in q and k. The depth of the descent tree is in $O(\log k)$ as with each application of the algorithm the degree is approximately divided by 2. The total number of nodes is less than or equal to its arity raised to the power of its depth, so $(q^2k)^{O(\log k)}$. Since any polynomial in q and k is absorbed in the $O()$ notation, we find that, assuming the two heuristics, any discrete logarithm in $K = \mathbb{F}_{q^{2k}}$, provided that there exists a representation of K in the form used above, can be computed in a time bounded by

$$\max(q, k)^{O(\log k)}.$$

Below, we discuss the consequences of this complexity for different types of finite fields $\mathbb{F}_Q$ where we want to compute the discrete logarithm. The complexities are expressed in terms of $\log Q$, which is the size of the input.

Case 1. In the optimal case, the finite field admits a δ-sparse medium subfield representation, so in particular $\mathbb{F}_Q = \mathbb{F}_{q^{2k}}$, and q and k are almost equal (or, more general, $k \leq q + O(1)$). Then, the complexity becomes $q^{O(\log q)}$. Since $Q \approx q^{2q}$, we have $q = (\log Q)^{O(1)}$, which gives an expression of the form $2^{O((\log\log Q)^2)}$. In this case the total algorithm is quasi-polynomial.

Case 2. The next case we consider is the case where the characteristic p is polynomial in the input size, so p is bounded by $(\log Q)^{O(1)}$. Let $n = \log Q / \log p$, so that $Q = p^n$. If p is too small or n does not factor nicely so that the field cannot be written in the form of case 1, we embed the discrete logarithm problem in $\mathbb{F}_Q$ into a discrete logarithm problem in a larger field.
To do so, let $k = n$ if n is odd, or $k = n/2$ if n is even. Then, set $q = p^{\lceil \log_p k \rceil}$. Now the field $\mathbb{F}_{q^{2k}}$ contains $\mathbb{F}_Q$ (because $p|q$ and $n|2k$), so we're back in case 1. In terms of Q we get a complexity $\log_p(Q)^{O(\log\log Q)}$, which can again be written as $2^{O((\log\log Q)^2)}$ (although the constant hidden in the $O()$ is of course larger here).

Case 3. The last case we consider is the case where $q = L_{q^{2k}}[\alpha]$. Here, the characteristic of the base field is not so small compared to the extension degree. The algorithm does not keep its quasi-polynomial complexity. Consider finite fields of the form $\mathbb{F}_Q = \mathbb{F}_{q^{2k}}$, where q grows slower than $L_Q[\alpha]$ for some constant α. In the worst case equality holds, so $q = L_Q[\alpha]$. Then we can write $\log q = O((\log Q)^\alpha (\log\log Q)^{1-\alpha})$, so $k = \log Q / \log q = O((\log Q / \log\log Q)^{1-\alpha})$. In particular, $k \leq q + O(1)$, hence we get a complexity of $q^{O(\log k)}$. The algorithm in this case has a subexponential running time $L_Q[\alpha]^{O(\log\log Q)}$.
This complexity is smaller than $L_Q[\beta]$ for any $\beta > \alpha$. Hence, for any $\alpha < 1/3$, this algorithm is faster than the FFS and its variants.

Support of heuristic 7.2. This heuristic is based on the fact that the probability that the left-hand side L_m yields a relation (i.e., is 1-smooth) is constant. Using this, it can be heuristically shown that the matrix $H(X)$ is expected to have $\Theta(q^3)$ rows, where the implicit constant depends only on $\delta = \max(\deg h_0, \deg h_1)$. Therefore, we have a system with $\Theta(q^3)$ relations between $O(q^2)$ unknowns. It seems reasonable to expect that it has full rank, even though the only way to support this heuristic is by testing on several inputs. This is done in e.g. [Jou14].

Support of heuristic 7.3. For a line of justification of heuristic 7.3, [BGJT14] shows the following. Denote by $\mathcal{H}$ the matrix of all $\#P_q = q^3 + q$ vectors $v(m)$ defined as in section 7.2. The matrix $H(P)$ is formed of the rows $v(m)$ such that the numerator of L_m is smooth. Again, it can be heuristically shown that the matrix $H(P)$ is expected to have $\Theta(q^3)$ rows, where the implicit constant depends only on δ. Now, the following proposition can be proven (for proof, see [BGJT14]).

Theorem 7.4. *Let l be a prime not dividing $q^3 - q$. Then the matrix $\mathcal{H}$ over $\mathbb{F}_l$ has full rank $q^2 + 1$.*

Although this result is encouraging, it of course does not prove that the submatrix $H(P)$ of $\mathcal{H}$ also has full rank. However, empirical results in [BGJT14] seem to indicate that this holds in practice.

7.4 Special cases, adaptions and open questions

In this section, some small improvements of the new index calculus algorithm are discussed, together with some open questions regarding this new algorithm.

Kummer extensions. Many discrete logarithm records over finite fields of characteristic 2 achieved by using the new method as treated in section 7.2 make use of Kummer extensions. A Kummer extension of a field K of degree n is an extension $K(\alpha) \supset K$ with $\alpha^n \in K$ and, in addition, K^* has to contain an element of order n. Suppose we work in a finite field of characteristic 2 of the form $\mathbb{F}_{q^{2k}}$, where $k = q - 1$. Let g be a generator of $\mathbb{F}_q^*$. If we set $h_0(X) = gX$ and $h_1(X) = 1$, then we see:

$$h_1(X)X^q - h_0(X) = X^q - gX = X(X^{q-1} - g),$$

so $I(X) = X^{q-1} - g$ is an irreducible factor of degree k (the proof of this is given in appendix A). Hence, $\mathbb{F}_{q^{2k}} \cong \mathbb{F}_{q^2}[X]/(I(X))$. This kind of an extension is called a *Kummer extension*. Note that the action of Frobenius on linear elements in this case becomes:

$$(X + \theta)^q = X^q + \theta^q \equiv g(X + \theta^q/g) \mod I(X).$$

This maps $X + \theta$ to a homography of $X + \theta$, hence we have a simple linear relation between the elements of the smoothness basis. This way, the size of the factor base is reduced.
Likewise, for the case $k = q + 1$, a *twisted Kummer extension* can be used. Let $h_0(X) = -G^{q-1}$ for a generator G of the multiplicative group $\mathbb{F}_{q^2}^*$ and $h_1(X) = X$, then we get $I(X) = X^{q+1} + G^{q-1}$ (the proof that $I(X)$ is irreducible is given in appendix A). Now, a similar result holds for the action of Frobenius, again reducing the size of the factor base.

Different construction of the extension. An alternative option for constructing the extension field proposed in [GKZ14] is the requirement that $I(X)$ is a degree k irreducible factor of:

$$h_1(X^q)X - h_0(X^q),$$

where h_1 and h_0 are again low degree polynomials. The advantage of this option is that it allows a wider range of possible extension degrees k for a given basefield $\mathbb{F}_{q^n}$ (note that before $n = 2$ was taken, as we needed an extension of $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$ is the smallest one. However, sometimes choosing $n > 2$ might be optimal for the algorithm). Let $\delta = \max(\deg h_0, \deg h_1)$. The former method contrains k to satisfy $k \leq q + \delta$. With this new method, we can reach k up to $k \leq q\delta + 1$.

Descent methods. In practice, the descent method as proposed in section 7.2 is not always used. Depending on the specific case, efficient descent principles can be combined. For example, one can start with a continued fraction descent, followed by a classical descent. These steps are comparable to the descent method in section 6.5, but due to some adaptions that need to be made to apply the classical descent in this algorithm, this method cannot descend to very low degrees in case of the new index calculus algorithm ([Jou14]).
Hence, if elements are of a low enough degree, the so-called *Gröbner basis descent* may be used. The description of this descent method can be found in [Jou14], and the idea is the following: given a polynomial $P(X) \in \mathbb{F}_{q^{2k}} \cong \mathbb{F}_{q^2}[X]/(I(X))$ (so again the extension is made using a degree k irreducible factor $I(X)$ of $h_1X^q - h_0$), we search for a pair of polynomials of low degree $g_1(X), g_2(X) \in \mathbb{F}_{q^{2k}}$ such that $P(X)$ divides $g_1(X)^q g_2(X) - g_1(X)g_2(X)^q$ mod $I(X)$. Then, we consider the relation:

$$g_1(X)^q g_2(X) - g_1(X)g_2(X)^q \equiv g_1(X)^q g_2(X) - g_1(X)g_2(X)^q \mod I(X).$$

Now, the right-hand side contains a factor equal to $P(X)$ and has a total degree of a small multiple (depending on the degrees of h_0 and h_1, as we used here $X^q \equiv \frac{h_0(X)}{h_1(X)} \mod I(X)$) of $D_m = \max(\deg g_1, \deg g_2)$. The left-hand side factors as $\prod_{(a,b)\in S}(\beta g_1 - \alpha g_2)$ (in the notation of section 7.2), so

this has factors of degree at most D_m. With good probability, we obtain a relation between $P(X)$ and polynomials of degree at most D_m, hence starting a descent procedure until only elements of the factor base are involved. The question is thus how to construct those g_1 and g_2. This is done using a Gröbner basis algorithm, as explained in [Jou14].

Open questions. There are still many open questions about the new index calculus algorithm. For example, can the heuristic hypotheses that are used in the algorithm be removed? Or, can the complexity be brought down to a polynomial time algorithm? Can the method (keeping its quasi-polynomial nature) be extended to finite fields with larger characteristic?
Considering recent developments, the question arises if the security of small characteristic parameters appearing in cryptographic systems is decreased. However, the latest records as published in [Jou14] and [ZGK14] using the new algorithm make use of (twisted) Kummer extensions, which simplify some computation steps in the algorithm hence reducing the complexity. In literature on cryptography, such specific structures are not considered. Some recent papers start to address these problems (see e.g. [GKZ14], [AMORH13]), but it is not clear whether the new techniques weaken existing cryptographic systems.

7.5 Example

We compute the discrete logarithms in $\mathbb{F}_{q^{2k}} = \mathbb{F}_{2^{4\cdot 2\cdot 15}} \cong \mathbb{F}_{2^{4\cdot 2}}[X]/(X^{15}+X+1)$ of all linear elements $X + v$, with $v \in \mathbb{F}_{q^2} = \mathbb{F}_{2^{4\cdot 2}}$, using the algorithm described above.

Setup. Let $q = p^n = 2^4$. Now $\mathbb{F}_q$ can be represented by $\mathbb{F}_2[U]/(U^4+U^3+1)$. The extension field $\mathbb{F}_{q^2}$ is represented by $\mathbb{F}_q[V]/(V^2+V+U)$. The k-th degree extension is made by letting $h_0 = X^2 + X$ and $h_1 = 1$, such that $h_1(X)X^q - h_0(X) = X^q - h_0(X)$ has an irreducible factor of degree 15, namely $I(X) = X^{15} + X + 1$. So $\mathbb{F}_{q^{2k}} \cong \mathbb{F}_{q^2}/(I(X))$. As a primitive element in $\mathbb{F}_{q^{2k}}$ to which base the discrete logarithms are computed we use $g = X+V$.

Finding relations. Now we seek $a, b, c, d \in \mathbb{F}_{q^2}$ with $ad \neq bc$ and a, b, c and d not all in $\mathbb{F}_q$, and consider the right-hand side of equation (7.3), which is a third degree polynomial:

$$(ca^q - ac^q)Xh_0(X) + (da^q - bc^q)h_0(X) + (cb^q - ad^q)X + (db^q - bd^q) \mod I(X),$$

and seek values a, b, c and d such that this factors into linear terms. For example, we found $a = V^{19}, b = V^{155}, c = V^{92}, d = V^{89}$ which satisfy $ad - bc \neq 0$ (as a shorthand notation, we write elements in $\mathbb{F}_{q^2}$ as a power of

V). Then the RHS of (7.3) becomes $X^3 + V^{136}X^2 + V^{17}$, which splits nicely into three linear factors, namely:

$$X^3 + V^{136}X^2 + V^{17} \equiv (X + V^{15})(X + V^{17})(X + V^{240}) \mod I(X)$$

In this case the LHS of equation (7.3) factors into 17 linear terms as follows:

$$\begin{aligned} X^{17} \quad + \quad & V^{102}X^{16} + V^{102}X + V^{17} = \\ & (X + V^{20})(X + V^{45})(X + V^{52})(X + V^{62})(X + V^{65})(X + V^{67}) \cdot \\ & (X + V^{73})(X + V^{124})(X + V^{136})(X + V^{148})(X + V^{199})(X + V^{205}) \cdot \\ & (X + V^{207})(X + V^{210})(X + V^{220})(X + V^{227})(X + V^{252}). \end{aligned}$$

In total, we see:

$$(X^3 + V^{136}X^2 + V^{17})(X^{17} + V^{102}X^{16} + V^{102}X + V^{17})^{-1} \equiv 1 \mod I(X).$$

So we have found a relation between 20 linear terms.

Linear algebra. If we have found enough (i.e. $> q^2$) relations, we can put it in a matrix M. Let the i-th column corresponds to the power of $X + V^i$, making the total of columns 256. Each relation can be put in a row by putting the power to which $X + V^i$ appears in the relation in the corresponding column.
When solving this matrix modulo $\text{ord}(g) = 2^{4 \cdot 2 \cdot 15} - 1 = 2^{120} - 1$, we expect to find the unique non-zero solution vector v to $vM^T \equiv 0 \mod 2^{120} - 1$, such that the i-th entry satisfies $g^i = X + V^i$. As the primitive element $g = X + V$ is an element of the factor base, we need the first entry of v, corresponding to the logarithm of $X + V$, to be 1. In our case, we do not get a unique solution because $2^{120} - 1$ factors as:

$$2^{120} - 1 = 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot (\text{larger primes}),$$

and those small primes turn out to give problems when solving the matrix. Indeed, the rank of M modulo 3, 5, 7 respectively is 252, 252 and 253, while we need rank 255 to solve the system (for all other primes in the factorization, the rank is good). If the rank is $q^2 - 1$, then the nullity is one (since there are q^2 columns), i.e. there is only one non-trivial solution. If the nullity is greater than one, there are more solutions and the answer you get may be a linear combination of those, which doesn't reveal the right answer. So, we solve modulo $G = (2^{120} - 1)/(3^2 \cdot 5^2 \cdot 7)$, then we know the solution up to a constant, i.e. $g^{n \cdot G + v_i} = X + V^i$ for some $0 \leq n \leq 3^2 \cdot 5^2 \cdot 7$, where v_i is the i-th entry of the vector v. All that is left to do is find this n for each i and then the logarithms of elements in the factor base are known. For example, we find:

$$g^{405648192073033408478945025720 32} = X + V^2.$$

Chapter 8

Conclusion

The most efficient algorithms to solve the discrete logarithm problem in finite field, i.e. $a^k = b$ in $\mathbb{F}_Q$ where $a, b \in \mathbb{F}_Q$ and Q is a prime or a prime power, are the index calculus algorithms. They consist of three main steps: a sieving step to create linear relations involving elements of the factor base, a linear algebra step to solve these relations and an individual logarithm step which can sometimes be included in the linear algebra step to solve for the discrete logarithm of b.
With the number field sieve, a number field $\mathbb{Q}(\alpha)$ is used to create elements such that for $a, b \in \mathbb{Z}$ and a specific integer m, $a + bm$ and the norm of $a + b\alpha$ are B-smooth for a fixed smoothness bound B. This algorithm runs in $L_Q[1/3, (\frac{64}{9})^{1/3}]$ for finite fields $\mathbb{F}_Q$ with $Q = p$ for p a large prime or $Q = p^n$, where p^n is a power of a medium to large prime.
The function field sieve uses two function fields to create relations. This works optimal for $\mathbb{F}_Q$ where $Q = p^n$ is a large power of a small or medium sized prime. In this case the algorithm runs in time $L_Q[1/3, (\frac{32}{9})^{1/3}]$.
In 2013/2014, a new index calculus algorithm is introduced, which, although relying on several heuristics, reaches a quasi-polynomial running time for $Q = q^{2k}$, where q is a power of a small prime, and $q \approx k$. In almost all other cases in small characteristic, this algorithm is faster than the function field sieve.
This last algorithm gives rise to many questions: can the heuristics in the algorithm be removed? Does this break-through affect the safety of cryptographic systems which make use of discrete logarithms in small characteristic? And, of course the most important question: can the complexity be brought down to polynomial time? In the case of small characteristic, with the new quasi-polynomial algorithm, this seems more plausible than ever, while in medium and large characteristic there is still a long way to go.

Appendix A

Some proofs

A.1 Proof: $f'(\alpha)O_K \subseteq \mathbb{Z}[\alpha]$ from section 5.2

We prove the following statement:

Theorem A.1. *Given a number field $K = \mathbb{Q}(\alpha)$, with α integer over $\mathbb{Z}$, its ring of integers O_K, and $f(x) \in \mathbb{Z}[x]$ the minimal polynomial of α. Then:*

$$f'(\alpha)O_K \subseteq \mathbb{Z}[\alpha],$$

where f' denotes the derivative of f.

Proof. Let $\mathbb{Z}[\alpha] = C$ and define:

$$C^* = \{x \in L : \ \mathrm{Tr}(xy) \in \mathbb{Z} \ \forall y \in C\}.$$

Here, Tr denotes the field trace, Tr: $\mathbb{Q}(\alpha) \to \mathbb{Q}$. Let δ denote the degree of $f(x)$. The next result follows from [Ser79, *Prop.* 3.11]:

$$C^* = \mathbb{Z} + \mathbb{Z} \cdot \frac{\alpha}{f'(\alpha)} + \cdots + \mathbb{Z} \cdot \frac{\alpha^{\delta-1}}{f'(\alpha)}.$$

Hence, $f'(\alpha)C^* \subseteq \mathbb{Z}[\alpha]$. If we prove that $O_K \subseteq C^*$, then:

$$f'(\alpha)O_K \subseteq f'(\alpha)C^* \subseteq \mathbb{Z}[\alpha].$$

To prove that $O_K \subseteq C^*$, take $\beta \in O_K$. Then we can prove that $\mathrm{Tr}(\beta \cdot c) \in \mathbb{Z}$ if $c \in C$. Since $c \in O_K$ and $\beta \in O_K$ it follows that $\beta \cdot c \in O_K$. We can show that if $\delta \in O_K$, then $\mathrm{Tr}(\delta) \in \mathbb{Z}$: the map of multiplication by δ is a map $\mathbb{Q}(\alpha) \to \mathbb{Q}(\alpha)$ whose basis over $\mathbb{Q}$ is $\mathbb{Q}$-linear. Write $[\delta] = (a_{ij})$, then $\mathrm{Tr}(\delta) = \sum a_{ii}$. Since a basis of O_K over $\mathbb{Z}$ is also a basis of $\mathbb{Q}(\alpha)$ over $\mathbb{Q}$ and for this last one we know that all $a_{ij} \in \mathbb{Z}$, we see that $\mathrm{Tr}(\delta) \in \mathbb{Z}$, so $O_K \subseteq C^*$. □

A.2 Proof of irreducibility of Kummer extensions from section 7.4

In this section, we will prove the statements in section (7.4): when $\mathbb{F}_{q^2}$ is a finite field of characteristic 2, then $f(X) = X^{q-1} - g$ with g a generator of $\mathbb{F}_q^*$ and $h(X) = X^{q+1} - G^{q-1}$ with G a generator of $\mathbb{F}_{q^2}^*$, are irreducible in $\mathbb{F}_{q^2}$. Although not mentioned in [Jou14], these statements are only true in characteristic 2. In the case of $f(X)$, this is proven in the proof. Both proofs make use of the following lemma:

Lemma A.2. Given a group G and a positive integer m, suppose $\alpha \in G$ is such that α^m has order m. Then $\text{ord}(\alpha) = m^2$.

Proof. Consider the subgroup of G generated by α. Then the map:

$$\begin{array}{rccc} \phi: & <\alpha> & \longrightarrow & <\alpha^m> \\ & \beta \in <\alpha> & \longmapsto & \beta^m \end{array}$$

is a surjective homomorphism, hence $\text{ord}(\alpha) = nm$ for some $n \in \mathbb{Z}_{>0}$. Also, $\alpha^{m^2} = (\alpha^m)^m = 1$, hence $n|m$ and in particular, $nm \leq m^2$.
The kernel of ϕ, $\text{Ker}(\phi)$, is a subgroup of $<\alpha>$, hence it is cyclic. We see $\alpha^n \in \text{Ker}(\phi)$ has order m, so $\text{Ker}(\phi)$ contains $\geq m$ elements. Thus:

$$m^2 \geq nm = \text{ ord}(\alpha) = \#\text{ Ker}(\phi) \cdot \#<\alpha^m> \quad \geq m^2$$

So indeed, $\text{ord}(\alpha) = m^2$. □

Now, we will prove the two statements.

Theorem A.3. *Let q be a prime power. The polynomial $f(X) = X^{q-1} - g$ with g a generator of $\mathbb{F}_q^*$, is irreducible in $\mathbb{F}_{q^2}$ only if char$(\mathbb{F}_{q^2}) = 2$.*

Proof. First, we see that if $f(X)$ is irreducible in $\mathbb{F}_q$, then it is not irreducible over $\mathbb{F}_{q^2}$ if $\text{char}(\mathbb{F}_q)$ is odd, since then $q-1$ is even. Then the extension $\mathbb{F}_q[\alpha]$ with α a zero of $f(X)$ is of even order $q-1$, so it has $\mathbb{F}_{q^2}$ as a subfield.
If $\text{char}(\mathbb{F}_q) = 2$, then $q-1$ is odd, so we see that $f(X)$ is irreducible over $\mathbb{F}_q$ if and only if it is irreducible over $\mathbb{F}_{q^2}$. Hence, proving that $f(X)$ is irreducible over $\mathbb{F}_q$ in any characteristic proves the theorem.
Consider a root α of $f(X)$. Using lemma $(A.2)$ we see that α has order $(q-1)^2$. We need to show that the index $[\mathbb{F}_q[\alpha] : \mathbb{F}_q] = q-1$. Say, $[\mathbb{F}_q[\alpha] : \mathbb{F}_q] = m$ for some $m \in \mathbb{Z}_{>0}$, then m satisfies $(q-1)^2 = q^m - 1$, so $m = \text{ ord}(q \mod (q-1)^2)$ in the group $(\mathbb{Z}/(q-1)^2\mathbb{Z})^*$.
Consider the map:

$$\pi : (\mathbb{Z}/(q-1)^2\mathbb{Z})^* \quad \longrightarrow \quad (\mathbb{Z}/(q-1)\mathbb{Z})^*.$$

In particular this map sends $\bar{q} \mapsto 1$ so $q \mod (q-1)^2$ is in $\text{Ker}(\pi)$. For $\text{Ker}(\pi)$ we see: $\text{Ker}(\pi) = \{1 + a(q-1) \mod (q-1)^2\} \cong \mathbb{Z}/(q-1)\mathbb{Z}$. Under

this isomorphism, $q \mapsto\in \mathbb{Z}/(q-1)\mathbb{Z}$ has order $q-1$, hence $m = q-1$ which proves that $f(X)$ is irreducible over $\mathbb{F}_q$ in any characteristic. □

Theorem A.4. *Let q be a prime power and let $\mathbb{F}_{q^2}$ be a finite field of characteristic 2. The polynomial $h(X) = X^{q+1} - G^{q-1}$ with G a generator of $\mathbb{F}^*_{q^2}$ is irreducible in $\mathbb{F}_{q^2}$.*

This proof is similar to the proof of the previous theorem.

Proof. Let α be a root of $h(X)$. By lemma $(A.2)$, α has order $(q+1)^2$. We need to show that $[\mathbb{F}_q[\alpha] : \mathbb{F}_q] = q+1$. This index equals the smallest n for which we have $(q+1)^2|(q^{2n}-1)$. So we want $(q^2)^n \equiv 1 \mod (q+1)^2$. We consider the following map:

$$\chi : (\mathbb{Z}/(q+1)^2\mathbb{Z})^* \quad \longrightarrow \quad (\mathbb{Z}/(q+1)\mathbb{Z})^*,$$

and in particular, $\bar{q^2} \longmapsto (q \mod (q+1))^2 = 1$ so $q^2 \mod (q+1)^2$ is in $\mathrm{Ker}(\chi)$, which is isomorphic to $\mathbb{Z}/(q+1)\mathbb{Z}$. Under the action of the kernel, we see that $q^2 \mapsto \overline{q-1} = -2$. As $q+1$ is odd, the order of -2 in ($\mathbb{Z}$ mod $(q+1)\mathbb{Z}$) equals $q+1$. Hence, $h(X)$ is irreducible in $\mathbb{F}_{q^2}$. □

Here, we see that if the characteristic is odd, then $q+1$ is even. The order of -2 in $\mathbb{Z} \mod (q+1)\mathbb{Z}$ is the smallest m such that $q+1|-2m$, so in this case the order is $(q+1)/2$, meaning that $h(X)$ is not irreducible when $\mathrm{char}(\mathbb{F}_q) \neq 2$.

Bibliography

[Adl94] L.M. Adleman. The function field sieve. *Algorithmic Number Theory, Proceedings of the ANTS-I conference*, 877:108–121, 1994.

[AH99] L.M. Adleman and M.-D. Huang. Function field sieve method for discrete logarithms over finite fields. *Information and Computation*, 151(1-2):5–16, 1999.

[AMORH13] G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Weakness of $\mathbb{F}_{3^{6\cdot 509}}$ for discrete logarithm cryptography. *Pairing-based cryptography-Pairing 2013*, 8365:20–44, 2013.

[AMORH14] G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Computing discrete logarithms in $\mathbb{F}_{3^{6\cdot 137}}$ and $\mathbb{F}_{3^{6\cdot 163}}$ using Magma. *Arithmetic of Finite Fields, Lecture Notes in Computer Science*, 9061:3–21, 2014.

[BBD+14] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H Jeljeni, E. Thomé, M. Videau, and P. Zimmermann. Discrete logarithms in GF(2^{809}) with FFS. *Public-Key Cryptography PKC 2014, Lecture Notes in Computer Science*, 8383:221–238, 2014.

[Ber70] E.R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24:713–735, 1970.

[BFO81] J. Brillhart, M. Filaseta, and A. Odlyzko. On an irreducibility theorem of A. Cohn. *Canadian Journal of Mathematics*, 33(5):1055–1059, 1981.

[BGJT14] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. *Advances in Cryptology, EUROCRYPT 2014, Lecture Notes in Computer Science*, 8441:1–16, 2014.

[BLP93] J.P. Buhler, H.W. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. *The development of the*

number field sieve, Lecture Notes in Mathematics, 1554:50–94, 1993.

[BP98] R.L. Bender and C. Pomerance. Rigorous discrete logarithm computations in finite fields via smooth polynomials. *AMS/IP Studies in Advanced Mathematics*, 7:221–232, 1998.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[Dic30] K. Dickman. On the frequency of numbers containing prime factors of a certain relative magnitude. *Arkiv för Mathematik, Astronomi och Fysik*, 22A(10):1–14, 1930.

[GGMZ13] F. Göloglu, R. Granger, G. McGuire, and J. Zumbrägel. On the function field sieve and higher splitting probabilities. *Advances in Cryptology - CRYPTO 2013, Lecture Notes in Computer Science*, 8043:109–128, 2013.

[GKZ14] R. Granger, T Kleinjung, and J. Zumbrägel. Breaking '128-bit secure' supersingular binary curves. *Advances in Cryptology - CRYPTO 2014, Lecture Notes in Computer Science*, 8617:126–145, 2014.

[Gor93] D.M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM Journal of Discrete Mathematics*, 6(1):124–138, 1993.

[HPS08] J. Hoffstein, J. Pipher, and J.H. Silverman. *An introduction to mathematical cryptography.* Springer, 2008.

[JL02] A. Joux and R. Lercier. The function field sieve is quite special. *Algorithmic Number Theory, Proceedings of the ANTS-V conference, Lecture Notes in Computer Science*, 2369:431–445, 2002.

[JL03] A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. *Mathematics of Computation*, 72:242:953–967, 2003.

[JL06] A. Joux and R. Lercier. The function field sieve in the medium prime case. *Advances in Cryptology - EUROCRYPT 2006, Lecture Notes in Computer Science*, 4004:177–193, 2006.

[JLSV06] A. Joux, R. Lercier, N. Smart, and F. Vercauteren. The number field sieve in the medium prime case. *Advances in Cryptology - CRYPTO 2006, Lecture Notes in Computer Science*, 4117:326–344, 2006.

[Jou13] A. Joux. Faster index calculus for the medium prime case. Application to 1175-bit and 1425-bit finite fields. *Advances in Cryptology - EUROCRYPT 2013, Lecture Notes in Computer Science*, 7881:254–270, 2013.

[Jou14] A. Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in small characteristic. *Selected Areas in Cryptography - SAC 2013, Lecture Notes in Computer Science*, 8282:355–379, 2014.

[JP14] A. Joux and C. Pierrot. Improving the polynomial time precomputation of Frobenius representation discrete logarithm algorithms. *Advances in Cryptology - ASIACRYPT 2014, Lecture Notes in Computer Science*, 8873:378–397, 2014.

[Kle06] T. Kleinjung. On polynomial selection for the general number field sieve. *Mathematics of Computation*, 75(256):2037–2047, 2006.

[Len90] H.W. Lenstra. Algorithms for finite fields. *Number Theory and Cryptography, London Mathematical Society Lecture Note Series*, 154:76–85, 1990.

[LL93] A.K. Lenstra and H.W. Lenstra. *The development of the number field sieve, Lecture Notes in Mathematics*, volume 1554. Springer, 1993.

[MOV01] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[Pol93] J.M. Pollard. The lattice sieve. *In [LL93]*, pages 43–49, 1993.

[Sch93] O. Schirokauer. Discrete logarithms and local units. *Philosophical Transansactions of the Royal Society of London, Series A*, 345(1676):409–423, 1993.

[Sch00] O. Schirokauer. Using number fields to compute logarithms in finite fields. *Mathematics of Computation*, 69(231):1267–1283, 2000.

[Sch02] O. Schirokauer. The special function field sieve. *SIAM Journal of Discrete Mathematics*, 16(1):81–98, 2002.

[Sch08] O. Schirokauer. The impact of the number field sieve on the discrete logarithm problem in finite fields. *Algorithmic Number Theory*, 44:397–420, 2008.

[Ser79] J.P. Serre. *Local Fields*. Springer, 1979.

[Ste12] P. Stevenhagen. *Number Rings*. Universiteit Leiden, websites.math.leidenuniv.nl/algebra/ant.pdf, 2012.

[Sti09] H. Stichtenoth. *Algebraic Function Fields and Codes, second edition*. Springer, 2009.

[Stu02] C. Studholme. The discrete log problem. *University of Toronto, www.cs.toronto.edu/∼cvs/dlog/research_paper.pdf (research paper)*, 2002.

[Tei98] J. Teitelbaum. Euclid's algorithms and the Lanczos method over finite fields. *Mathematics of Computation*, 67(224):1665–1678, 1998.

[Web96] D. Weber. Computing discrete logarithms with the general number field sieve. *Algorithmic Number Theory, Proceedings of the ANTS-II conference, Lecture Notes in Computer Science*, 1122:391–403, 1996.

[ZGK14] J. Zümbragel, R. Granger, and T. Kleinjung. Discrete logarithms in GF(2^{9234}). *NMBRTHRY list, listserv.nodak.edu/archives/nmbrthry*, 31 Jan. 2014.