NTNU
Norwegian University of
Science and Technology

# The Number Field Sieve for Discrete Logarithms

Henrik Røst Haarberg

Master of Science
Submission date: June 2016
Supervisor: Kristian Gjøsteen, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

**Abstract**

We present two general number field sieve algorithms solving the discrete logarithm problem in finite fields. The first algorithm presented deals with discrete logarithms in prime fields $\mathbb{F}_p$, while the second considers prime power fields $\mathbb{F}_{p^n}$. We prove, using the standard heuristic, that these algorithms will run in sub-exponential time.

We also give an overview of different index calculus algorithms solving the discrete logarithm problem efficiently for different possible relations between the characteristic and the extension degree.

To be able to give a good introduction to the algorithms, we present theory necessary to understand the underlying algebraic structures used in the algorithms. This theory is largely algebraic number theory.

# Contents

# 1 Introduction

## 1.1 Discrete logarithms

Consider the situation where we have numbers $k$, $g$ and $t$ such that $t^k = g$ in a finite group (that is, in the finite group $G$ with a cyclic subgroup $H$ with generator $t$, $g \in H$ and $k \in \mathbb{Z}$). There exist fast algorithms for finding $g$ given $k$ and $t$ (for example, exponentiation by squaring). On the other hand, finding $k$ given $g$ and $t$ is not easy in general. Doing this computation will in this text be referred to as solving the discrete logarithm problem. This is usually written as $k = \log_t g$, mirroring standard logarithm notation, in for example $\mathbb{R}$.

## 1.2 The general number field sieve and $L$-notation

This text will discuss variants of the general number field sieve algorithm used to solve discrete logarithm problems in finite fields of both prime and prime power order. That is, the text will consider the discrete logarithm problem over finite fields $\mathbb{F}_p$ and $\mathbb{F}_{p^n}$.

The discrete logarithm problem is important in cryptography, where the Diffie-Hellman, Elgamal and Digital Signature algorithms are based on the assumption that the discrete logarithm problem is hard. It can in fact be shown that there exists no sub-exponential algorithm for general groups. To be sub-exponential in solving the discrete logarithm problem, the algorithm needs to take the structure of the group into consideration. In the case where the underlying group is a finite field, the general number field sieve is known to solve the problem in sub-exponential time, but still super-polynomial time. To compare asymptotic run times $L$-notation will be used,

$$L_Q(\alpha, c) = \exp((c + o(1))(\log Q)^\alpha (\log \log Q)^{1-\alpha}), \tag{1}$$

where $Q$ will be the size of the finite field we consider. This expression is sub-exponential in the number of bits $b$ required to express $Q$. That is, $L$-notation will look like

$$L_Q(\alpha, c) = \exp((c + o(1))b^\alpha (\log b)^{1-\alpha}), \tag{2}$$

if we consider it a function of $b = \log Q$. In this notation we see that $\alpha = 1$ corresponds to a fully exponential run time, while $\alpha = 0$ corresponds to a

polynomial run time. The two primary algorithms we will consider in this text will have $\alpha = 1/3$ and $c = (64/9)^{1/3}$.

The general number field sieve algorithm was first developed to factor large integers. It was later modified to solve discrete logarithms, in finite fields, both of prime order and prime power order. The general number fields sieve is the asymptotically fastest algorithm to solve the discrete logarithm problem, and it is also practically the fastest when the size of the field gets large enough.

Much of the literature explains the discrete logarithm algorithm by comparing it to the factorization algorithm. This will not be done here. This text will first include prerequisites to understand the algorithms, then an examination of the algorithm in the prime field case, an overview of different algorithms that solves the discrete logarithm problem in different finite fields, before it finishes with an algorithm that works for certain prime power finite fields. This thesis follows in the footsteps of several earlier master theses studying the number field sieve conducted at NTNU [10, 13, 14].

# 2 Theory

This text assumes some knowledge about basic abstract algebra, e.g. understanding of rings, fields, polynomials and modules. We will assume that the rings in this text are commutative and unital.

## 2.1 Number fields

The general number field sieve algorithm will make use of number fields in computing the discrete logarithm. In this section we will define what that is and consider related subjects. We will also give some other theory which will be necessary to understand the algorithms presented.

**Definition 1.** A *number field* is a finite degree field extension of the field of rational numbers $\mathbb{Q}$. Here its dimension as a vector space over $\mathbb{Q}$ is simply called its degree.

Typically we consider a monic irreducible polynomial $f \in \mathbb{Q}[x]$ of degree $d$, with root $\alpha \in \mathbb{C}$. Then the field extension

$$\mathbb{Q}[\alpha] = \{a_0 + a_1\alpha + \ldots + a_{d-1}\alpha^{a-1} \mid a_i \in \mathbb{Q} \ \forall i\}$$

is a number field of degree $d$. We can map elements of a number field into the complex numbers by way of an embedding.

**Definition 2.** An *embedding* of a number field $\mathbb{Q}[\alpha]$ is an injective homomorphism $\mathbb{Q}[\alpha] \to \mathbb{C}$. The number of such embeddings is equal to the degree, $d$. An embedding is called *real* if the image is a subset of $\mathbb{R}$, and *complex* if not.

Inside a number field we have a notion of integer mirroring the usual integers $\mathbb{Z}$.

**Definition 3.** An element $\beta \in \mathbb{Q}[\alpha]$ is an *algebraic integer* if it is a root of a monic polynomial with integer coefficients, that is, $\beta$ is an algebraic integer if it is a root of monic $g(x) \in \mathbb{Z}[x]$.

The collection of algebraic integers form a ring called the ring of (algebraic) integers, denoted $\mathcal{O}_{\mathbb{Q}[\alpha]}$. Note that $\mathbb{Z} \subseteq \mathcal{O}_{\mathbb{Q}[\alpha]}$ in any number field. We now have two different notions of integers, and to differentiate elements in $\mathbb{Z}$ and $\mathcal{O}_{\mathbb{Q}[\alpha]}$ it is typical to call $\mathbb{Z}$ the set of rational integers. A number that is both rational and an algebraic integer is automatically a rational integer,

$(w \in \mathbb{Q} \wedge w \in \mathcal{O}_{\mathbb{Q}[\alpha]} \Rightarrow w \in \mathbb{Z})$, giving justification for the name rational integer. It should be noted that the ring $\mathcal{O}_{\mathbb{Q}[\alpha]}$ is not in general equal to $\mathbb{Z}[\alpha]$. It is in fact the case that $\mathbb{Z}[\alpha] \subseteq \mathcal{O}_{\mathbb{Q}[\alpha]}$. To give an example of $\mathbb{Z}[\alpha] \neq \mathcal{O}_{\mathbb{Q}[\alpha]}$, consider the number field $\mathbb{Q}[\sqrt{-3}]$. The ring of integers in this number field is called the Eisenstein integers. To see that $\mathcal{O}_{\mathbb{Q}[\sqrt{-3}]} \neq \mathbb{Z}[\sqrt{-3}]$, consider the number $\frac{-1+\sqrt{-3}}{2} = e^{2\pi i/3}$ which we will call $\omega$. It is clear that $\omega \notin \mathbb{Z}[\sqrt{-3}]$, but we have $\omega \in \mathcal{O}_{\mathbb{Q}[\sqrt{-3}]}$ since we have $\omega^2 + \omega + 1 = 0$, so it is a root of a monic polynomial in $\mathbb{Z}[x]$. It is possible to show that $\mathcal{O}_{\mathbb{Q}[\sqrt{-3}]} = \mathbb{Z}[\omega]$.

To use the ring of integers in the number field sieve algorithm, we need to know some properties of the ring. Most of the properties we will use are covered by the fact that the ring of integers is a Dedekind domain. (See Marcus[9, page 56] for a proof.)

### 2.1.1 Dedekind domains

Informally, a Dedekind domain is a ring where unique factorization holds for ideals in the ring, but not necessarily for the elements themselves. An ideal $\mathfrak{i}$ factors into another ideal $\mathfrak{p}$ if it can be written as $\mathfrak{i} = \mathfrak{pq}$ where $\mathfrak{q}$ is another ideal. This clearly mirrors factorization in $\mathbb{Z}$.

**Definition 4.** A *Dedekind domain* is an integral domain, where every nonzero proper ideal factors into prime ideals.

A Dedekind domain is therefore a ring with no zero-divisors, where all ideals (not equal to zero or the whole ring) can be expressed as a product of prime ideals, that is we have the unique factorization theorem for ideals, which, like we said, mirrors the unique factorization in $\mathbb{Z}$.

It is important to emphasize that this theorem is not true for elements in $\mathcal{O}_{\mathbb{Q}[\alpha]}$, only ideals. For example in $\mathcal{O}_{\mathbb{Q}[\sqrt{-5}]} = \mathbb{Z}[\sqrt{-5}]$ the element $6 = 2 \cdot 3 = (1 + \sqrt{-5})(1 - \sqrt{-5})$ has two factorizations which are different. On the other hand, the ideal $\langle 6 \rangle$ has a unique factorization, $\langle 6 \rangle = \langle 2, 1 + \sqrt{-5} \rangle^2 \langle 3, 1 - \sqrt{-5} \rangle \langle 3, 1 + \sqrt{-5} \rangle$. (See Marcus [9] for more details.) This is because the ideals $\langle 2 \rangle = \langle 2, 1 + \sqrt{-5} \rangle^2$ and $\langle 3 \rangle = \langle 3, 1 + \sqrt{-5} \rangle \langle 3, 1 - \sqrt{-5} \rangle$ factor non-trivially into prime ideals. Note that all factors of $\langle 6 \rangle$ are generated by two elements, it is in fact true in a Dedekind domain that all ideals are generated by one or two elements.

A fact that we will use later is that all prime ideals are maximal ideals in a Dedekind domain. The converse is always true, so in a Dedekind domain an ideal is prime if and only if it is maximal.

Unique factorization of elements holds in unique factorization domains (UFDs). A Dedekind domain is a UFD if and only if it is a PID, a ring where all ideals are generated by a single element. In fact, we have a notion of "how much" unique factorization fails in a Dedekind domain, called the ideal class group. To define this, we need to look at some other related definitions, namely the field of fractions of an integral domain and fractional ideals.

**Definition 5.** A *field of fractions* of an integral domain is the smallest field in which it can be embedded.

The canonical example is that the field of fractions of $\mathbb{Z}$ is $\mathbb{Q}$. The field of fractions of $\mathbb{Z}[\sqrt{-5}]$ is equal to $\mathbb{Q}[\sqrt{-5}]$.

**Definition 6.** Let $R$ be a Dedekind domain and $K$ be its field of fractions. Consider a $R$-submodule of $K$ that we call $\mathfrak{a}$. Let $\mathfrak{a}$ be nonzero. This $\mathfrak{a}$ is a *fractional ideal* if there exists a nonzero $r \in R$ such that $r\mathfrak{a} \subset R$.

Note that a fractional ideal does not lie in $R$, but rather in $K$. One can think of $r$ as "clearing out denominators" in $\mathfrak{a}$. Given a Dedekind domain $R$, we now define a relation $\sim$ on the fractional ideals of $R$ such that $\mathfrak{a} \sim \mathfrak{b}$ if $r\mathfrak{a} = s\mathfrak{b}$ for $r, s \in R$. This relation can be shown to be an equivalence relation.

**Theorem 7.** Multiplication of equivalence classes of ideals defined by $[\mathfrak{a}][\mathfrak{b}] = [\mathfrak{a}\mathfrak{b}]$ is well defined, and the classes form a group under this operation.

*Proof.* Assume $\mathfrak{a}_1 \sim \mathfrak{b}_1$ and $\mathfrak{a}_2 \sim \mathfrak{b}_2$, that is $r_1\mathfrak{a}_1 = s_1\mathfrak{b}_1$ and $r_2\mathfrak{a}_2 = s_2\mathfrak{b}_2$. Multiplying these equations together, we get

$$\begin{aligned} r_1\mathfrak{a}_1 r_2\mathfrak{a}_2 &= s_1\mathfrak{b}_1 s_2\mathfrak{b}_2 \\ r_1 r_2\mathfrak{a}_1\mathfrak{a}_2 &= s_1 s_2\mathfrak{b}_1\mathfrak{b}_2 \\ \mathfrak{a}_1\mathfrak{a}_2 &\sim \mathfrak{b}_1\mathfrak{b}_2, \end{aligned}$$

so the multiplication is well defined. The group operation is commutative and associative because multiplication of ideals is commutative and associative. The identity is the class of all principle ideals. The existence of inverses are proven in Marcus [9, chapter 3]. □

**Definition 8.** The *ideal class group* of a Dedekind domain is the group given by defining multiplication between the equivalence classes of ideals.

We note that the group is trivial if and only if $R$ is a PID (and hence a UFD), since all ideals will then be in the same equivalence class, the one consisting of principal ideals. We noted that $\mathbb{Z}[\sqrt{-5}]$ did not have unique factorization, and can hence not have trivial ideal class group. The ideal class group of $\mathbb{Z}[\sqrt{-5}]$ is in fact $\mathbb{Z}_2$, as is proven in Jamroz [5].

### 2.1.2 Module structure

A fact that will be used in the discussion of the algorithm is the free module structure of rings of integers. In a similar fashion to how $\mathbb{Q}[\alpha]$ is a vector space of degree $d$ over $\mathbb{Q}$, $\mathcal{O}_{\mathbb{Q}[\alpha]}$ is a free $\mathbb{Z}$-module of degree $d$.

### 2.1.3 Norm of ideals

Given unique factorization of ideals in $\mathcal{O}_{\mathbb{Q}[\alpha]}$, we want another way to look at these ideals, which will be the norm of them. First we consider the norm of an element in a number field. Remember that in a number field $\mathbb{Q}[\alpha]$ of degree $d$, we have exactly $d$ embeddings into $\mathbb{C}$, namely the homomorphisms defined by $\sigma_i : \alpha \mapsto \alpha_i$ where $\alpha_1, ..., \alpha_d$ are the roots of $f$. To define the norm of an element we use these embeddings. The norm of an element $\beta \in \mathbb{Q}[\alpha]$ is

$$N(\beta) = \prod_{i=1}^{d} \sigma_i(\beta).$$

The norm sends elements in $\mathbb{Q}[\alpha]$ to $\mathbb{Q}$, and elements in $\mathcal{O}_{\mathbb{Q}[\alpha]}$ to $\mathbb{Z}$. We also see from the definition that the norm is multiplicative, i.e. $N(\beta\gamma) = N(\beta)N(\gamma)$. For elements $a + b\alpha \in \mathbb{Z}[\alpha]$ we can make the norm explicit,

$$N(a + b\alpha) = \prod_{i=1}^{d} \sigma_i(a + b\alpha) = \prod_{i=1}^{d} (a + b\alpha_i) = b^d \prod_{i=1}^{d} \Big(\frac{a}{b} + \alpha_i\Big).$$

Notice that we have $f(x) = \prod_{i=1}^{d}(x - \alpha_i)$, and that we are only a sign away from this. We continue

$$N(a + b\alpha) = (-b)^d \prod_{i=1}^{d} \Big( - \frac{a}{b} - \alpha_i\Big) = (-b)^d f\Big( - \frac{a}{b}\Big). \tag{3}$$

Furthermore we define the norm for ideals by $N(\mathfrak{a}) = [\mathcal{O}_{\mathbb{Q}[\alpha]} : \mathfrak{a}] = |\mathcal{O}_{\mathbb{Q}[\alpha]}/\mathfrak{a}|$. This definition has a close relationship with the norm of elements,

in fact for a principle ideal $\mathfrak{b}$ generated by $\beta$ we have that $N(\mathfrak{b}) = |N(\beta)|$. The norm of ideals is also multiplicative, $N(\mathfrak{a} \cdot \mathfrak{b}) = N(\mathfrak{a})N(\mathfrak{b})$ if $\mathfrak{a}$ and $\mathfrak{b}$ are ideals of $\mathcal{O}_{\mathbb{Q}[\alpha]}$.

Another important concept related to this is *ramification* of primes in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. Given the factorization of the ideal $\langle q \rangle$ generated by a prime $q$

$$\langle q \rangle = \prod_i \mathfrak{q}_i^{e_i}$$

into prime ideals $\mathfrak{q}$, we say that $q$ is *unramified* in $\mathcal{O}_{\mathbb{Q}[\alpha]}$ if $e_i \in \{0, 1\}$ for all $i$, and say it is *ramified* otherwise. The largest $e_i$ is called the ramification index of $q$ in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. Looking at the example above we see that 2 is ramified in $\mathcal{O}_{\mathbb{Q}[\sqrt{-5}]}$ because $\langle 2 \rangle = \langle 2, 1 + \sqrt{-5} \rangle^2$, while 3 is unramified in $\mathcal{O}_{\mathbb{Q}[\sqrt{-5}]}$, because $\langle 3 \rangle = \langle 3, 1 + \sqrt{-5} \rangle \langle 3, 1 - \sqrt{-5} \rangle$.

### 2.1.4 Units

In our discussion of the algorithms, we also encounter units of a ring i.e. elements with multiplicative inverse. Recall that all rings will have a unit, 1, as it is its own multiplicative inverse. All elements in a field, except zero, are units. The set of units of a ring is a group under multiplication, called the unit group. In a ring of integers in a number field, we have a theorem called Dirichlet's unit theorem which tell us about the structure of the unit group.

**Theorem 9** (Dirichlet's unit group theorem)**.** Let $\mathcal{O}^*_{\mathbb{Q}[\alpha]}$ be the unit group of the ring of integers $\mathcal{O}_{\mathbb{Q}[\alpha]}$, $r_1$ the real embeddings of the number field and $r_2$ the number of conjugate pairs of complex embeddings. Then $\mathcal{O}^*_{\mathbb{Q}[\alpha]} \cong G \times \mathbb{Z}^{r_1+r_2-1}$ where $G = \langle u_0 \rangle$ is a finite cyclic group consisting of all the roots of unity in $\mathbb{Q}[\alpha]$.

*Proof.* A proof can be found in Ash [1, Section 6.2]. □

## 2.2 Prime ideals

In the number field sieve algorithm we want to have some control over certain prime ideals in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. First note that if we have an ideal $\mathfrak{q}$ such that $N(\mathfrak{q})$ is prime, then $\mathfrak{q}$ is a prime ideal. This is because if $N(\mathfrak{q}) = q$ for a prime $q$, then $|\mathcal{O}_{\mathbb{Q}[\alpha]}/\mathfrak{q}| = q \implies \mathcal{O}_{\mathbb{Q}[\alpha]}/\mathfrak{q} \cong \mathbb{Z}_q$, which is a field. This means that $\mathfrak{q}$ is a maximal ideal and hence a prime ideal. Conversely we have that any prime ideal $\mathfrak{q}$ has prime power norm, since prime ideals are maximal in a

Dedekind domain, which means that $\mathcal{O}_{\mathbb{Q}[\alpha]}/\mathfrak{q}$ is a finite field, which implies $N(\mathfrak{q}) = |\mathcal{O}_{\mathbb{Q}[\alpha]}/\mathfrak{q}| = q^n$ as any finite field has prime or prime power order. We use this power $n$, and say that such an ideal has degree $n$. We are interested in first degree prime ideals.

To identify first degree prime ideals we first define the set

$$R(q) = \{r \in \{0, 1, ..., q\} \mid f(r) \equiv 0 \pmod{q}\}, \tag{4}$$

where $f$ is the polynomial we used to define our number field. We look at what this set has to do with factors of $N(a + b\alpha)$.

**Theorem 10.** Let $q$ be a prime number and $a + b\alpha \in \mathbb{Z}[\alpha]$ such that $q$ does not divide $b$. Then $q \mid N(a + b\alpha)$ if and only if $a \equiv -br \pmod{q}$ for a $r \in R(q)$.

*Proof.* Let $r \in R(q)$ be such that $a \equiv -br \pmod{q}$. We now simply calculate the norm,

$$N(a + b\alpha) \equiv (-b)^d f\Big(-\frac{a}{b}\Big) \equiv (-b)^d f\Big(-\frac{-br}{b}\Big) \equiv (-b)^d f(r) \equiv 0 \pmod{q}.$$

If $q \mid N(a+b\alpha)$, we have $(-b)^d f(-\frac{a}{b}) \equiv 0 \pmod{q}$, which means $f(-\frac{a}{b}) \equiv 0 \pmod{q}$ since $b \nmid q$. Define $r = -\frac{a}{b}$ to get what we want.

□

We now classify first degree prime ideals.

**Theorem 11.** There is a bijection between pairs $(q, r)$ such that $r \in R(q)$ and first degree prime ideals $\mathfrak{q}$ of $\mathbb{Z}[\alpha]$.

*Proof.* To find a map from first degree prime ideals to such pairs, first note that the definition gives that the norm of a first degree prime ideal is a prime $q$, so we send the ideal to this $q$ in the pair. Next, consider the natural homomorphism $\phi : \mathbb{Z}[\alpha] \to \mathbb{Z}_q$, and let $\phi(\alpha) = r$. This $r$ will in fact be in $R(q)$. This because $f(\alpha) = 0$ giving us $\phi(f(\alpha)) = 0$. We also have $\phi(f(\alpha)) \equiv f(r) \pmod{q}$. To see this we simply distribute $\phi$ over $f$,

$$\phi(f(\alpha)) = \phi\left(\prod_{i=0}^{d} c_i \alpha^i\right) \equiv \prod_{i=0}^{d} c_i \phi(\alpha)^i \equiv \prod_{i=0}^{d} c_i r^i \equiv f(r) \pmod{q},$$

giving us $f(r) \equiv 0 \pmod{q}$.

To find the reverse map we consider the same homomorphism $\phi : \mathbb{Z}[\alpha] \to \mathbb{Z}_q$. We now study $\ker \phi$ which is equal to the cyclic subgroup of $\mathbb{Z}[\alpha]$ generated by $a+b\alpha$ ($\ker \phi = \langle a+b\alpha \rangle$). The map $\phi$ is surjective, so the first isomorphism theorem for rings says that $\mathbb{Z}[\alpha]/\ker \phi \cong \mathbb{Z}_q$. If we define $\mathfrak{q} = \ker \phi$, we have that $N(\mathfrak{q}) = q$, so $\mathfrak{q}$ is a first degree prime ideal.

Note that $\mathfrak{q}$ is generated by $q$ and $\alpha - r$. We have

$$\begin{aligned}\phi(\beta_1 q + \beta_2(\alpha - r)) &= \phi(\beta_1)\phi(q) + \phi(\beta_2)\phi(\alpha) - \phi(\beta_2)\phi(r) \\ &= \phi(\beta_1) \cdot 0 + \phi(\beta_2)r - \phi(\beta_2)r \\ &= 0,\end{aligned}$$

showing that $\{\beta_1 q + \beta_2(\alpha - r) | \beta_1, \beta_2 \in \mathbb{Z}[\alpha]\} \subset \ker \phi = \mathfrak{q}$. To show the other inclusion, we assume $\beta \in \mathfrak{q}$, which is the same as $\phi(\beta) \equiv 0 \pmod{q}$. Any $\beta \in \mathbb{Z}[\alpha]$ can be written as $\beta = \sum_{i=0}^{d-1} a_i \alpha^i$. There is an integer $k$ such that

$$\begin{aligned}kq = \phi(\beta) &= \phi\left(\sum_{i=0}^{d-1} a_i \alpha^i\right) = \sum_{i=0}^{d-1} a_i r^i \\ 0 &= kq - \sum_{i=0}^{d-1} a_i r^i \\ \beta &= kq - \sum_{i=0}^{d-1} a_i r^i + \beta \\ \beta &= kq - \sum_{i=0}^{d-1} a_i r^i + \sum_{i=0}^{d-1} a_i \alpha^i = kq - \sum_{i=0}^{d-1} a_i(\alpha^i - r^i).\end{aligned}$$

We observe that $(\alpha - r)$ is a factor of the sum, which means that we have proved the second inclusion.

We now need to show that if $\mathfrak{q}_1 = \langle q, \alpha - r_1 \rangle$ and $\mathfrak{q}_2 = \langle q, \alpha - r_2 \rangle$, then either $\mathfrak{q}_1 \neq \mathfrak{q}_2$ and $r_1 \not\equiv r_2 \pmod{q}$; or $\mathfrak{q}_1 = \mathfrak{q}_2$ and $r_1 \equiv r_2 \pmod{q}$. We consider $(\alpha - r_1) - (\alpha - r_2) = r_2 - r_1$. Equality trivially gives us $\mathfrak{q}_1 = \mathfrak{q}_2$, so we consider $r_2 - r_1 \neq 0$. We consider $\gcd(r_2 - r_1, q)$, which can equal 1 or $q$.

First, if $\gcd(r_2 - r_1, q) = q$, then $r_2 - r_1 = (\alpha - r_1) - (\alpha - r_2) = kq$ with $k \in \mathbb{Z}$. This means that $\alpha - r_1 = \alpha - r_2 + kq$, so we get $\mathfrak{q}_1 = \mathfrak{q}_2$.

Now, if $\gcd(r_2 - r_1, q) = 1$, we assume $\mathfrak{q}_1 = \mathfrak{q}_2$ for a contradiction. This means that $q$, $\alpha - r_1$ and $\alpha - r_2$ are all in the ideal, and thus 1 is also in the ideal. This is a contradiction because then the ideal would equal the whole ring. We conclude that $\mathfrak{q}_1 \neq \mathfrak{q}_2$ and $r_1 \not\equiv r_2 \pmod{q}$. □

Continuing our example in $\mathbb{Z}[\sqrt{-5}]$, we try this with the prime ideals $\langle 3, 1+\sqrt{-5}\rangle$ and $\langle 3, 1-\sqrt{-5}\rangle$. Note that $\mathbb{Z}[\sqrt{-5}]$ has minimal polynomial $f(x) = x^2 + 5$. First we find that the ideals have the same norm, which is 3. Then we consider $R(3) = \{1, 2\}$. We define our homomorphism by $\sqrt{-5} \mapsto 2 \pmod 3$, which gives $-\sqrt{-5} \mapsto -2 \equiv 1 \pmod 3$. That is we have the following representations

$$\langle 3, 1+\sqrt{-5}\rangle \longleftrightarrow (3, 2)$$
$$\langle 3, 1-\sqrt{-5}\rangle \longleftrightarrow (3, 1).$$

We check if $\langle 3, 1+\sqrt{-5}\rangle$ is generated by 3 and $\sqrt{-5} - 2$.

$$\begin{aligned}\langle 3, \sqrt{-5} - 2\rangle &= 3k + \gamma(\sqrt{-5} - 2) \\ &= 3n + 3\gamma + \gamma(\sqrt{-5} - 2) \\ &= 3n + \gamma(1 + \sqrt{-5}) \\ &= \langle 3, 1 + \sqrt{-5}\rangle\end{aligned}$$

by choosing $k = n + \gamma$. We also see that the prime ideal $\langle 3, 1-\sqrt{-5}\rangle$ is generated by 3 and $\sqrt{-5} - 1 = -(1 - \sqrt{-5})$.

## 2.3 Smooth numbers

**Definition 12.** A *y-smooth* integer has all prime factors less than or equal to $y$.

That means that $15 = 3 \cdot 5$ is 5-smooth, while $14 = 2 \cdot 7$ is not. This notion can be extended to elements of $\mathcal{O}_{\mathbb{Q}[\alpha]}$ by defining $\beta \in \mathcal{O}_{\mathbb{Q}[\alpha]}$ to be $y$-smooth if its norm $N(\beta) \in \mathbb{Z}$ is $y$-smooth.

### 2.3.1 Density

Denote by $\psi(x, y)$ the number of integers less than $x$ which are $y$-smooth. The probability of a random integer in $1, ..., x$ being $y$-smooth is then $\frac{\psi(x,y)}{x}$.

### 2.3.2 Exponent vectors

An alternative way to express $y$-smooth numbers given their factorizations is by their exponent vectors. Given a base $q_1, q_2, ...q_{\pi(y)}$ (typically called a

smoothness base) that consists of all prime numbers up to $y$, one writes the number $n$ as a vector

$$n \mapsto e_{q_1}(n), e_{q_2}(n), ..., e_{q_{\pi(y)}}(n)$$

where $e_{q_i}(n)$ is the exponent of the $i$th prime in the factorization of $n$. E.g., given the basis (2, 3, 5) for 5-smooth numbers, $24 = 2^3 \cdot 3$ is written $(3, 1, 0)$ while $25 = 5^2$ is written $(0, 0, 2)$. This base for $y$-smooth numbers will be called the rational factor base $\mathcal{R}$. It should be noted that it is possible to have a rational factor base where one considers not only $y$-smooth numbers, but e.g. a factor base consisting of $y$-smooth numbers for a small $y$ and a large prime $q > y$.

It is also possible to create an algebraic factor base, $\mathcal{A}$. This will consist of first degree prime ideals in $\mathcal{O}_{\mathbb{Q}[\alpha]}$, represented by the pair $(q, r)$. Then the exponent vector representing the element $(a + b\alpha) \in \mathbb{Z}[\alpha]$ will be representing the factorization of the ideal $\langle a + b\alpha \rangle$ into prime ideals.

One should note that it is possible to expand the algebraic factor bases, $\mathcal{A}$. In both algorithms, we will let $\mathcal{A}$ consist of first degree prime ideals represented by a pair $(q, r)$. There is another type of prime ideal that could divide $a + b\alpha$, namely prime ideals that divide the index $f_\alpha = [\mathcal{O}_{\mathbb{Q}[\alpha]} : \mathbb{Z}[\alpha]]$, although these two options cover all possibilities. These extra ideals complicate the exposition, and was for that reason excluded. To find out how to include and treat these ideals, check Joux et.al. [8, Section 4.1].

# 3 The number field sieve in prime fields

## 3.1 Overview

In our finite field $\mathbb{F}_p$ we consider $t, g \in \mathbb{F}_p^*$, such that $g \in \langle t \rangle$. The goal is to find the discrete logarithm of $g$ to the base $t$, $\log_t g$. To compute this logarithm using the general number field sieve we first find discrete logarithms modulo large prime divisors $l$ of $p-1$, that is we find relations of the form $j \equiv \log_t g \pmod{l}$, and use the Chinese remainder theorem to find the original logarithm $\log_t g$. Finding the relations modulo small $l$ is done with algorithms which are faster in small groups, and will not be covered here.

Note that the prime factorization of $p-1$ is typically known in most cryptographic contexts, and even if it is not, it is possible to use the factorization version of the number field sieve to find it, which runs in the same time as this algorithm.

To find such relations for large $l$ we first choose two number rings with maps into $\mathbb{F}_p$. That is, we first need two polynomials $f_1, f_2 \in \mathbb{Z}[x]$ with a common root.

There are several options for number rings here. The most common is probably to use the base $m$ technique where one simply chooses a number $m$ and takes the base-$m$ expansion of $p = \sum a_i m^i$. This gives $f_1 = \sum a_i X^i$, which has a common root $m$ with $f_2 = X - m$. The number ring we get from $f_2$ is then isomorphic with $\mathbb{Z}$, this is called the rational side, while the side corresponding to $f_1$ is called algebraic.

Another option is to take a degree $d+1$ polynomial with small coefficients, and try to find a root of it modulo $p$, discarding it and trying again if no root is found. When such a polynomial is found, it is possible to use lattice techniques to find a degree $d$ polynomial with the same root. (See Joux and Lercier [6] for details.)

This search for good polynomials defining the number rings is an active field of research, with no known optimal solution.

## 3.2 Calculating logarithms

We will continue this section with the base $m$ method, from the possible methods of choosing polynomials. Other choices of polynomials give similar continuations, only with two algebraic sides, rather than one rational and one algebraic side. We do not explicitly define the polynomial on the rational

side, and simply call the base $m$ polynomial $f$. To find a degree $d$ polynomial $f$, we start by defining $m = \lfloor p^{1/d} \rfloor$, then we find the base $m$ expansion of $p = \sum a_i m^i$ and define

$$f = \sum_{i=0}^{d} a_i X^i.$$

We see that $f(m) \equiv 0 \pmod{p}$. Our polynomial is also monic ($a_d = 1$) because $2m^d > p > m^d$.

In practice, it may be interesting to try to find different polynomials so that we can find a polynomial where the coefficients $a_i$ are small, as the size of these coefficients change the run time. Therefore, it may be worthwhile to check different degrees $d$ and hope for small coefficients. We will return to the choice of $d$ in the section about the run time of the algorithm where we will find bounds on $d$.

The next step is to find a root $\alpha \in \mathbb{C}$ of $f$, and thus defining $\mathbb{Z}[\alpha]$ and $\mathcal{O}_{\mathbb{Q}[\alpha]}$. We will consider on one side elements in $\mathbb{Z}[\alpha]$. The other ring will simply be $\mathbb{Z}$, as discussed above. We also need homomorphisms from these rings into $\mathbb{F}_p$. The map from $\mathbb{Z}$ will be the canonical projection map, $\pi : \mathbb{Z} \to \mathbb{F}_p$ given by $a \mapsto a \pmod{p}$ and the map $\phi : \mathbb{Z}[\alpha] \to \mathbb{F}_p$ will be given by $\sum b_i \alpha^i \mapsto \sum b_i m^i \pmod{p}$ for $b_i \in \mathbb{Z}$, or equivalently the homomorphism given by $\alpha \mapsto m \pmod{p}$ .

The algorithm will try to find powers of $l$, with a certain form,

$$o^l = t^{x_t} g \prod_{(a,b)\in S} (a + bm)^{x_{a,b}} \tag{5}$$

$$\beta^l = \prod_{(a,b)\in S} (a + b\alpha)^{x_{a,b}}, \tag{6}$$

for $o \in \mathbb{Z}$ and $\beta \in \mathbb{Z}[\alpha]$. Using our homomorphisms, we get that

$$\pi(o^l) = \pi\left(t^{x_t} g \prod_{(a,b)\in S} (a + bm)^{x_{a,b}}\right) = \pi(t^{x_t} g)\pi\left(\prod_{(a,b)\in S} (a + bm)^{x_{a,b}}\right).$$

For $t, g \in \mathbb{F}_p$, we identify canonical integers, and by abuse of notation, we call these integers $t$ and $g$. Looking at the two parts of the right expression, we can see that $\pi(t^{x_t} g) = t^{x_t} g$ using our abuse of notation. We also see that

$$\pi\left(\prod_{(a,b)\in S} (a + bm)^{x_{a,b}}\right) = \phi\left(\prod_{(a,b)\in S} (a + b\alpha)^{x_{a,b}}\right)$$

because $\pi(m) = \phi(\alpha)$. This results in the relation $\pi(o^l) = t^{x_t} g\phi(\beta^l)$ for $x_t$. This relation can be used to get a discrete logarithm modulo $l$. To do this we see that since $\pi$ and $\phi$ are homomorphisms, $\pi(o^l) = \pi(o)^l$ and $\phi(\beta^l) = \phi(\beta)^l$ are $l$th powers. Renaming $c = \pi(o)(\phi(\beta))^{-1}$ we get

$$c^l = t^{x_t} g. \tag{7}$$

Since $t^{x_t} g \in \langle t \rangle$, we see that also $c \in \langle t \rangle$. This means that $c = t^b$ for some $b$. Inserting this into (7) we get $t^{bl} = t^{x_t} g$. Looking at the exponents we get

$$bl = x_t + \log_t g$$

in $\mathbb{F}_p^*$. In addition to being true modulo $p-1$, this equation will also be true modulo divisors $l$ of $p-1$, but since $bl \equiv 0 \pmod{l}$ the equation simplifies. We get that

$$\log_t g \equiv -x_t \pmod{l}, \tag{8}$$

a solution modulo $l$. If we can generate such solutions for every prime divisor of $p-1$, we can use the Chinese remainder theorem to get back the discrete logarithm $k \equiv \log_t g \pmod{p-1}$, solving the discrete logarithm problem.

## 3.3 Sieving

To find the relations in (5, 6), we first use a technique called sieving. We want to find a set of numbers that are smooth over some factor base, that is numbers that can be written as an exponent vector over such a base. First consider two sets of primes $\mathcal{R}$ and $\mathcal{A}$ called the rational and algebraic factor base, respectively.

Like we alluded to in the theory section, a common and effective way to find such a factor base is to simply consider all primes up to a certain bound $y$. We will assume that this approach is taken when we analyze the run time of the algorithm. The difference between these sets is that $\mathcal{A}$ also has an $r$ attached to each such prime. In other words: $\mathcal{A}$ consists of pairs $(q, r)$, the representatives of first degree prime ideals seen in (4). We are looking for numbers that are smooth over both $\mathcal{R}$ and $\mathcal{A}$. On the rational side this simply means numbers $a + bm$ that factor completely over $\mathcal{R}$. On the algebraic side we want numbers $a + b\alpha \in \mathbb{Z}[\alpha]$ such that $\langle a + b\alpha \rangle$ factors completely into first degree prime ideals that are represented by a pair $(q, r) \in \mathcal{A}$. In the end we want a set $S \subset \mathbb{Z} \times \mathbb{Z}$ such that for $(a, b) \in S$ both $a + bm$ and $a + b\alpha$ are smooth in their respective factor bases.

Let $v$ be a bound, whose size will be discussed when we cover the run time of the algorithm. First we look at the rational sieve. We fix $0 \leq b \leq v$, and list the values $a + bm$ for all $a$ such that $|a| \leq v$. A prime $q$ divides $a + bm$ if and only if $a \equiv -bm \pmod{q}$, so after calculating $-bm$, we check this congruence for all $a$ considered. If we find that $q$ divides $a + bm$, we update the number by dividing it by $q$ as many times as possible. Now, when all $a$ are checked, scan the array for entries with value $\pm 1$, as these will correspond to smooth numbers. Then go to the next $b$ and continue the procedure. Note that we skip $a$ if $a$ and $b$ are not coprime, as they give no new information.

The algebraic sieve is similar, first we find $N(a + b\alpha)$ for all possible $a$ with a fixed $b$. We check the congruence $a \equiv -br \pmod{q}$, and update the corresponding entries by dividing by the highest possible power of $q$. When this is done for all possible pairs $(q, r)$, we again check for entries that are equal to $\pm 1$, as these identify the smooth numbers. Remember that the algebraic factor base $\mathcal{A}$ consists of first degree prime ideals, and we are checking the factorization of ideals in $\mathcal{O}_{\mathbb{Q}[\alpha]}$.

To increase the speed of this step in the algorithm we can change division to subtraction by storing $\log(a + bm)$ instead of $a + bm$, and subtracting $\log q$ instead of dividing by $q$. Then we look for values that are close to zero when checking for smooth values. This can be improved to initializing by zeros, and adding $\log q$ instead of subtracting. We also store the exponent vectors of the smooth numbers for later use.

The last step is to create the set $S$ with all smooth numbers by taking the numbers that are smooth in both factor bases, the intersection of the results of both sieves. This set $S$ will be used later in the algorithm.

## 3.4 Schirokauer maps

The equation we want to get to on the algebraic side is

$$\beta^l = \prod_{(a,b) \in S} (a + b\alpha)^{x_{a,b}},$$

in other words we want $\beta^l$ to be equal something. Unfortunately, the algebraic factor base consists of ideals in $\mathcal{O}_{\mathbb{Q}[\alpha]}$, and if we use these directly, we only get a relation with $\langle \beta \rangle^l$, not with $\beta^l$ itself. We will in other words have

$$\langle \beta \rangle^l = \prod_{(a,b) \in S} (a + b\alpha)^{x_{a,b}},$$

from the algebraic factor base. To increase the probability of the right hand side actually being an $l$th power of elements in $\mathcal{O}_{\mathbb{Q}[\alpha]}$, and not just ideals, we introduce another condition that has to be true in if it is the case that the right hand side is an $l$th power of elements in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. We will use Schirokauer maps [12], and the condition we are considering is that the maps are zero.

Remember that $l$ is a prime divisor of $p-1$. We assume that $l$ is unramified in $\mathcal{O}_{\mathbb{Q}[\alpha]}$, which means that $l$ has no repeated factors in its factorization into prime ideals. Define the set

$$\Gamma = \{\gamma \in \mathcal{O}_{\mathbb{Q}[\alpha]} | N(\gamma) \not\equiv 0 \pmod{l}\},$$

the set of all algebraic numbers whose norm is not zero modulo $l$. We wish to find an integer $\epsilon$ such that $\gamma^\epsilon \equiv 1 \pmod{l}$ $\forall \gamma \in \Gamma$. We first consider the prime ideals $\mathfrak{l}_i$ dividing $\langle l \rangle$. For each such prime ideal $\mathfrak{l}_i$, let $\epsilon_{\mathfrak{l}_i} = |(\mathcal{O}_{\mathbb{Q}[\alpha]}/\mathfrak{l}_i)^*|$, and $\epsilon = \text{lcm}(\epsilon_{\mathfrak{l}_1}, ..., \epsilon_{\mathfrak{l}_k})$. Given this, we know that there exist $r_i$ such that $\epsilon = \epsilon_{\mathfrak{l}_i} r_i$ for all $i$. Now we consider

$$\gamma^\epsilon + \mathfrak{l}_i = (\gamma + \mathfrak{l}_i)^\epsilon = (\gamma + \mathfrak{l}_i)^{\epsilon_{\mathfrak{l}_i} r_i} = (\gamma + \mathfrak{l}_i)^{|(\mathcal{O}_{\mathbb{Q}[\alpha]}/\mathfrak{l}_i)^*| r_i}.$$

Since $\gamma \in \mathfrak{l}_i$, we are essentially raising $\gamma$ an exponent equal to the size of the multiplicative subgroup, making it equal 1. This leads to $\gamma^\epsilon + \mathfrak{l}_i = (1 + \mathfrak{l}_i)^{r_i} = 1 + \mathfrak{l}_i$, and since this is true for all $i$, we get that

$$\gamma^\epsilon - 1 \in \bigcap_i \mathfrak{l}_i.$$

The intersection of all $\mathfrak{l}_i$ is equal to the product of them, which is equal to $l\mathcal{O}_{\mathbb{Q}[\alpha]}$, since $l$ is unramified. Now we have that $\gamma^\epsilon - 1 \in l\mathcal{O}_{\mathbb{Q}[\alpha]}$ and hence $\gamma^\epsilon \equiv 1 \pmod{l}$.

Now define

$$\begin{aligned} \lambda : \Gamma &\to l\mathcal{O}_{\mathbb{Q}[\alpha]}/l^2\mathcal{O}_{\mathbb{Q}[\alpha]} \\ \gamma &\mapsto (\gamma^\epsilon - 1) + l^2\mathcal{O}_{\mathbb{Q}[\alpha]}. \end{aligned}$$

**Theorem 13.** The map $\lambda$ is a logarithmic homomorphism.

*Proof.*

$$\begin{aligned} &\lambda(\gamma\hat{\gamma}) - (\lambda(\gamma) + \lambda(\hat{\gamma})) \\ &\quad = (\gamma\hat{\gamma})^\epsilon - 1 - \gamma^\epsilon + 1 - \hat{\gamma}^\epsilon + 1 + l^2\mathcal{O}_{\mathbb{Q}[\alpha]} \\ &\quad = \gamma\hat{\gamma}^\epsilon - \gamma^\epsilon - \hat{\gamma}^\epsilon + 1 + l^2\mathcal{O}_{\mathbb{Q}[\alpha]} \\ &\quad = (\gamma^\epsilon - 1)(\hat{\gamma}^\epsilon - 1) + l^2\mathcal{O}_{\mathbb{Q}[\alpha]} = 0 + l^2\mathcal{O}_{\mathbb{Q}[\alpha]} \end{aligned}$$

since both $\gamma^\epsilon - 1$ and $\hat{\gamma}^\epsilon - 1$ are in $l\mathcal{O}_{\mathbb{Q}[\alpha]}$, the product of them is in $l^2\mathcal{O}_{\mathbb{Q}[\alpha]}$. $\square$

We now use the fact that $\mathcal{O}_{\mathbb{Q}[\alpha]}$ is a free $\mathbb{Z}$-module of rank $d$. This structure induces a structure on $l\mathcal{O}_{\mathbb{Q}[\alpha]}/l^2\mathcal{O}_{\mathbb{Q}[\alpha]}$, making it a free $\mathbb{Z}_l$ module of rank $d$. Given a basis $\{b_j l + l^2\mathcal{O}_{\mathbb{Q}[\alpha]}\}_{j=1,...,d}$ of this module, we create $d$ maps by projecting $\lambda$ into each coordinate. Call these maps $\lambda_j : \Gamma \to \mathbb{Z}_l$. Explicitly, we have

$$\gamma^\epsilon - 1 \equiv l \sum_{j=1}^{d} \lambda_j(\gamma) b_j \pmod{l^2}.$$

The point of these maps is that $\lambda_j(\beta^l) = l\lambda_j(\beta) = 0$ for all $j$, the maps are in other words zero for any $l$th power of elements in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. This will be used in the following section.

## 3.5 Linear algebra

Now, we want to use the relations we have gathered to find the relations (5, 6). In other words, we want a subset $U \subseteq S$ so that

$$o^l = t^{x_t} g \prod_{(a,b)\in S} (a + bm)^{x_{a,b}}$$

and

$$\beta^l = \prod_{(a,b)\in S} (a + b\alpha)^{x_{a,b}}.$$

This is done by linear algebra on the exponent vectors defining the numbers in $S$. This will result in a solution vector $x = (x_t, x_{(a,b)_1}, ..., x_{(a,b)_{|S|}})$, fulfilling the above relations.

We first consider how to construct the exponent vectors. Let $n(\mathcal{R})$ be the number of primes in the rational factor base $\mathcal{R}$, and let $n(\mathcal{A})$ be the number of elements in the algebraic factor base $\mathcal{A}$. We now construct an exponent vector $V_{(a,b)}$ for each element $(a, b) \in S$ with length $n(\mathcal{R}) + n(\mathcal{A}) + d$. The first $n(\mathcal{R})$ elements are the exponent vector in the rational factor base $\mathcal{R}$, the next $n(\mathcal{A})$ are from the algebraic factor base $\mathcal{A}$ and the last $d$ elements are from the Schirokauer maps. We also consider the numbers in the original problem $k = \log_t g$, and create exponent vectors $V_t = (e_{q_1}(t), ..., e_{q_{n(\mathcal{R})}}(t), 0, ..., 0, ..., 0)$ and $V_g = (e_{q_1}(g), ..., e_{q_{n(\mathcal{R})}}(g), 0, ..., 0, ..., 0)$. The first $n(\mathcal{R})$ elements are the factorizations of $t$ and $g$ in $\mathcal{R}$, and the remaining $n(\mathcal{A}) + d$ elements are zero.

Now we create a matrix $A = (V_t, V_{(a,b)_1}, ..., V_{(a,b)_{|S|}})$ and create the linear system we want to solve,

$$Ax \equiv -V_g \pmod{l}.$$

Note that almost all entries of $A$ are zero. This means that $A$ is a sparse matrix, and this is a fact we need to consider when solving this linear system. In fact, we cannot solve this linear system using Gaussian elimination, then the run time we want will not be archived (Gaussian elimination runs in $O(n^3)$.) Instead of using Gaussian elimination, we use an algorithm which fits much better for our problem, namely Wiedemann's algorithm [15]. We assume that the linear system is solvable. Note that if $l$ is large, it is unlikely that $l$ divides $\det(A)$, making is unlikely that $A$ is singular modulo $l$, as seen in Schirokauer [11]. Balancing the number of rows and columns is done in the run time section.

This will result in a solution vector $x = (x_t, x_{(a,b)_1}, ..., x_{(a,b)_{|S|}})$. We now define

$$z = t^{x_t} g \prod_{(a,b)\in S} (a + bm)^{x_{a,b}}$$
$$\delta = \prod_{(a,b)\in S} (a + b\alpha)^{x_{a,b}}.$$

To show that these can be written as $l$th powers, we use the properties that comes om the linear relations. We write these as

$$V_{q_i}(t)x_t + \sum_{(a,b)\in S} V_{q_i}(a + bm)x_{a,b} + V_{q_i}(g) \equiv 0 \pmod{l}$$
$$\sum_{(a,b)\in S} V_{Q_i}(a + b\alpha)x_{a,b} \equiv 0 \pmod{l}$$
$$\sum_{(a,b)\in S} \lambda_j(a + b\alpha)x_{a,b} \equiv 0 \pmod{l}$$

for $q_i \in \mathcal{R}$ and $Q_i \in \mathcal{A}$, with $\lambda_j$ denoting the Schirokauer maps. To see that we are dealing with $l$th powers, we replace $a + bm$, $a + b\alpha$, $t$ and $g$ with their representation in the smoothness bases,

$$z = \Big(\prod_i q_i^{V_{q_i}(t)}\Big)^{x_t} \prod_i q_i^{V_{q_i}(g)} \prod_{(a,b)\in S} \Big(\prod_i q_i^{V_{q_i}(a+bm)}\Big)^{x_{a,b}}$$
$$= \prod_i q_i^{V_{q_i}(t)x_t + V_{q_i}(g) + \sum_{(a,b)\in S} V_{q_i}(a+bm)x_{a,b}}.$$

We notice that $z$ is equal to something raised raised to an exponent which is equal to zero modulo $l$, which means that $z$ is an $l$th power. This calculation

will be exactly analogous for $\langle\delta\rangle$. The fact that the Schirokauer maps are logarithmic homomorphisms, gives

$$\lambda_j(\delta) = \lambda_j\Big(\prod_i (a+b\alpha)^{x_{a,b}}\Big) = \sum_{(a,b\in S)} x_{a,b}\lambda_j(a+b\alpha),$$

which is zero modulo $l$, which is unlikely to happen if $\delta$ is not an $l$th power in $\mathcal{O}_{\mathbb{Q}[\alpha]}$.

We can use this to write $z$ as an $l$th power $z = o^l$ and we also have that $\langle\delta\rangle = \langle\beta\rangle^l$, with the Schirokauer maps making it very probable that $\delta = \beta^l$ in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. So probable in fact, that we assume this to be the case.

We know that we can solve the discrete logarithm problem modulo $l$ from these equations. Then we repeat the algorithm for the other prime factors of $p-1$, and use the Chinese remainder theorem to get the logarithm we wanted to start with.

### 3.5.1 A note about smooth $t$ and $g$

One should note that finding $\log_t g$ in the above given algorithm depends on $t$ and $g$ to be smooth (to define $V_t$ and $V_g$). This is in fact not necessary for the algorithm to function. What we need is that $t$ and $g$ have smooth preimages under $\phi$. If this is needed, we define $\phi(\tau) = t$ and $\phi(\nu) = g$ such that the ideals generated by $\tau$ and $\nu$ are smooth. Replace $(V_t, V_g)$ by $(V_\tau, V_\nu)$, which are the exponent vectors connected to $\langle\tau\rangle$ and $\langle\nu\rangle$, and also update the Schirokauer maps $\lambda$. This gives the following relations form the linear algebra,

$$z = \tau^{x_\tau}\nu \prod_{(a,b)\in S} (a+bm)^{x_{a,b}}$$
$$\delta = \prod_{(a,b)\in S} (a+b\alpha)^{x_{a,b}}$$

which replace the normal relations.

To find out more about this detail, check Schirokauer [12].

If we cannot find smooth preimages of $t$, we have an alternative; if we find another generator $t'$ with a smooth preimage, we can find the logarithm of $g$ by

$$\log_t g \equiv \frac{\log_{t'} g}{\log_{t'} t} \pmod{p-1}$$

## 3.6 Run time

In this section we want to justify that this algorithm has a run time of

$$L_p(1/3, (64/9)^{1/3}) = \exp(((64/9)^{1/3} + o(1))(\log p)^{1/3}(\log\log p)^{2/3}),$$

it is in other words possible to find discrete logarithms in $\mathbb{F}_p$ within this bound. The lemmas presented here are found in Buhler et. al. [2]. Note that $o(1)$ is for when $p \to \infty$. Remember that the rational and algebraic factor bases $(\mathcal{R}, \mathcal{A})$ will be assumed to consist of the primes less than $y$, so that $n(\mathcal{R})$ is the number of primes less than or equal to $y$, $\pi(y)$, and the equivalent for $\mathcal{A}$. To give this algorithm a good run time, we need to balance the sieving step with the linear algebra step, as we need a certain amount of smooth numbers to solve the linear system, but we do not want to do the sieving step for longer than necessary.

To do this we first give a theorem for the density of smooth numbers and how it relates to $L$-notation.

**Theorem 14.** Given a function $g$ defined for all $y \geq 2$ that satisfies $g(y) \geq 1$ and $g(y) = y^{1+o(1)}$ for $y \to \infty$, we have

$$\frac{xg(y)}{\psi(x,y)} \geq L_x(1/2, \sqrt{2})$$

for $x \to \infty$. Also, equality is achieved precisely when $y = L_x(1/2, 1/\sqrt{2})$, for $x \to \infty$.

*Proof.* A proof can be found in Buhler et. al. [2, page 76]. □

In our algorithm we have certain parameters we can adjust to make sure it is as fast as possible. We have the degree $d$ of our polynomial $f$ and we have the bound of the sieving $v$. Remember that given $d$ we define $m = \lfloor p^{1/d} \rfloor$ and that $m$ is the bound of the coefficients in $f$. We are going to assume that $p > d^{2d^2}$, giving a restriction on $d$.

Before we determine the run time, we give some necessary lemmas, which can seem a bit technical, but which will be used to great effect later.

**Lemma 15.** Given the relation

$$\frac{v^2}{\log v} = av + b$$

for $a, b, v \geq e$, then it can be shown that

$$2v = (1 + o(1))\Big(a \log a + \sqrt{(a \log a)^2 + 2b \log b}\Big)$$

as $a + b \to \infty$.

*Proof.* A proof can be found in Buhler et. al. [2, page 78]. □

**Lemma 16.** For each pair $p, d \in \mathbb{N}$ of positive integers with the relation $p > d^{2d^2}$, the real numbers $u(p, d) \geq 2$ and $y(p, d) \geq 2$ are given, with the property that the real number

$$x(p, d) = 2dp^{2/d}u^{d+1}$$

satisfies the relation

$$\frac{u^2\psi(x, y)}{x} \geq g(y)$$

for some function $g(y) \geq 1$ with $g(y) = y^{1+o(1)}$ as $y \to \infty$. Then we have

$$2 \log u \geq (1 + o(1))\bigg(d \log d + \sqrt{(d \log d)^2 + 4 \log(p^{1/d}) \log \log(p^{1/d})}\bigg).$$

*Proof.* We have that $x^x > p$, and therefore $x \to \infty$ when $p \to \infty$. Now, use the last assumption combined with theorem 14 to get

$$u^2 \geq \frac{xg(y)}{\psi(x, y)} \geq L_x(1/2, \sqrt{2}).$$

In other words we have

$$u^2 \geq \exp\Big((\sqrt{2} + o(1))\sqrt{\log x \log \log x}.\Big)$$

Taking the logarithm and squaring we get

$$4(\log u)^2 \geq (2 + o(1)) \log x \log \log x,$$

dividing by two gives us

$$2(\log u)^2 \geq (1 + o(1)) \log x \log \log x.$$

Note that $\frac{z}{\log z}$ is an increasing function for $z > e$. Using this, we divide each side of the equation by its logarithm to get

$$\frac{2(\log u)^2}{\log 2 + 2 \log \log u} \geq (1 + o(1)) \frac{\log x \log \log x}{\log \log x + \log \log \log x}.$$

We can now cancel both $\log 2$ and $\log\log\log x$ to get

$$\frac{(\log u)^2}{\log\log u} \geq (1+o(1))\log x.$$

Using the definition of $x$ we get

$$\frac{(\log u)^2}{\log\log u} \geq (1+o(1))((2/d)\log p + (d+1)\log u).$$

We now realize we are in a position to apply Lemma 15 with $\log u = v$, $a \geq d+1$ and $b \geq (2/d)\log p$ to obtain the desired result of

$$2\log u \geq (1+o(1))\Big(d\log d + \sqrt{(d\log d)^2 + 4\log(p^{1/d})\log\log(p^{1/d})}\Big),$$

which concludes the proof. □

It is now time to start the analysis of the sieving step of the algorithm. We wish to find a lower bound for the time taken by this step. We know that for the linear algebra step to be successful, we need the number of rows in our matrix to be higher than the number of columns. The rows in our matrix is the number of smooth integers we find (in both the rational and algebraic factor base). We know that the probability that a random integer less than $x$ is $y$-smooth is $\frac{\psi(x,y)}{x}$. In the sieving step, we check around $v^2$ such integers. This means, if the integers we check in the sieve behave like they are random, we can expect to find $\frac{v^2\psi(x,y)}{x}$ $y$-smooth integers. There is no proof that this is true (we are not picking random integers), but we presume it is true and make the heuristic assumption that we find this many smooth integers.

The numbers of columns in the matrix considered in the linear algebra step is $n(\mathcal{R}) + n(\mathcal{A}) + d$, where both $n(\mathcal{R})$ and $n(\mathcal{A})$ is bound by $y$. The parameter $d$ is much smaller than $y$, as we will see later. This means that we can approximate the number of columns with $y$. Given these approximations, we get that having at least as many rows as columns is the same as

$$\frac{v^2\psi(x,y)}{x} \geq y.$$

We recognize this from Lemma 16, but to use that lemma we also need $x$ to be of a certain form, $x = 2dp^{2/d}u^{d+1}$, with $u$ replaced by $v$. We argue that all numbers we check are smaller or equal to this bound.

In our algorithm, we check $(a+bm)$ and $N(a+b\alpha)$ for smoothness, which is the same as checking $(a+bm)N(a+b\alpha)$. Recall that we only check $(a,b)$ if both integers are less than $v$, and also that the coefficients of $f$ are bound by $m = \lfloor p^{1/d} \rfloor$. We want to consider the largest $N(a+b\alpha)$ that we check.

First write

$$N(a+b\alpha) = (-b)^d f\Big(-\frac{a}{b}\Big).$$

This monic polynomial consists of $d+1$ therms, where the maximum of the first term is $v^d$, while the rest of the terms has a maximum of $mv^d$. The bound becomes $N(a+b\alpha) \leq v^d m(d+1)$. Giving us

$$\begin{aligned}(a+bm)N(a+b\alpha) &\leq (v+vm)v^d m(d+1)\\ &= v^{d+1}(m+1)m(d+1)\\ &\leq 2m^2 v^{d+1} d\\ &\leq 2dp^{2/d}v^{d+1}.\end{aligned}$$

Thus, we can set $x = 2dp^{2/d}v^{d+1}$ and apply Lemma 16. This gives us

$$2\log v \geq (1+o(1))\Big(d\log d + \sqrt{(d\log d)^2 + 4\log(p^{1/d})\log\log(p^{1/d})}\Big)$$

taking the exponential, we get

$$v^2 \geq (1+o(1))\exp\Big(\Big(d\log d + \sqrt{(d\log d)^2 + 4\log(p^{1/d})\log\log(p^{1/d})}\Big)\Big).$$

When we sieve, we do it in approximately $v^2$ steps, and the above is therefore a lower bound for this step. We now want to show that this lower bound is achievable with the right choices of $v$ and $y$. We choose $v$ to be the square root of the optimal run time, and also set $y$ equal to this number

$$v_0 = y_0 = \exp\Big(\frac{1}{2}\Big(d\log d + \sqrt{(d\log d)^2 + 4\log(p^{1/d})\log\log(p^{1/d})}\Big)\Big).$$

We also set $x_0 = 2dp^{2/d}v_0^{d+1}$. These choices gives us the relation

$$\frac{v_0^2\psi(x_0,y_0)}{x_0} = y_0^{1+o(1)}.$$

We want the above relation to be an inequality, which will be done by increasing $v$ and $y$ by $\epsilon > 0$

$$v = y = \exp\Big(\frac{1+\epsilon}{2}\Big(d\log d + \sqrt{(d\log d)^2 + 4\log(p^{1/d})\log\log(p^{1/d})}\Big)\Big),$$

with $x = 2dp^{2/d}v^{d+1}$. This means that $y = y_0^{1+\epsilon}$ and $v = v_0^{1+\epsilon}$, but $x \leq x_0^{1+\epsilon}$. Taking the logarithm, we get

$$\frac{\log x}{\log y} \geq \frac{(1+\epsilon)\log x_0}{(1+\epsilon)\log y_0} = \frac{\log x_0}{\log y_0}$$

Using this, we reach the an inequality involving $x$ and $x_0$

$$\frac{\psi(x,y)}{x} \geq \left(\frac{\psi(x_0,y_0)}{x_0}\right)^{1+o(1)}.$$

We now wish to use this inequality to generate an inequality involving $\frac{v^2\psi(x,y)}{x}$, the estimated count of smooth numbers we find, or equivalently, the number of rows in our matrix.

$$\begin{aligned}
\frac{v^2\psi(x,y)}{x} &\geq \left(\frac{v^2\psi(x_0,y_0)}{x_0}\right)^{1+o(1)} = \left(\frac{v_0^{2(1+\epsilon)}\psi(x_0,y_0)}{x_0}\right)^{1+o(1)} \\
&= \left(v_0^{2\epsilon}y_0^{1+o(1)}\right)^{1+o(1)} = \left(v_0^{2\epsilon}y_0\right)^{1+o(1)} \\
&= \left(y_0^{1+2\epsilon}\right)^{1+o(1)} = \left(y^{\frac{1+2\epsilon}{1+\epsilon}}\right)^{1+o(1)} \\
&> y^{1+o(1)}.
\end{aligned}$$

We want to compare the rows and columns in our linear system. Consider the number of columns in the linear system we solve in the algorithm, $n(\mathcal{R}) + n(\mathcal{A}) + d$. We know that $d$ is bounded by $\log p = y^{o(1)}$. Remember that $n(\mathcal{R})$ is the number of smooth numbers less than $y$, so $n(\mathcal{R}) \leq y$. We also have that

$$n(\mathcal{A}) \leq d \cdot n(\mathcal{R}) \leq n(\mathcal{R})\log p \leq y\log p = y^{1+o(1)}$$

since there is at most $d$ extensions of each number in $\mathcal{A}$ compared to $\mathcal{R}$. We get

$$n(\mathcal{R}) + n(\mathcal{A}) + d = y^{1+o(1)}$$

as the number of columns. This means that for these choices of $v$ and $y$ we have more rows than columns in our linear system, making it solvable. We let $\epsilon \to 0$ as $p \to \infty$ to get

$$v = y = \exp\left(\frac{1}{2} + o(1)\left(d\log d + \sqrt{(d\log d)^2 + 4\log(p^{1/d})\log\log(p^{1/d})}\right)\right).$$

In our expression for $v$ and $y$ we have a part containing $p$ and $d$. We want to minimize this with a good choice of $d$. In the expression

$$\exp\left(d\log d + \sqrt{(d\log d)^2 + 4\log(p^{1/d})\log\log(p^{1/d})}\right)$$

we assume that

$$d \approx \left(\frac{\log p}{\log\log p}\right)^{1/3},$$

which means that

$$\log d \approx \frac{1}{3}(\log\log p - \log\log\log p) \approx \log\log p.$$

We can then find the following

$$\begin{aligned} d\log d &\approx (\log p)^{1/3}(\log\log p)^{2/3} \\ (d\log d)^2 &\approx (\log p)^{2/3}(\log\log p)^{4/3} \\ \log p^{1/d} &\approx (\log p)^{2/3}(\log\log p)^{1/3} \\ \log\log p^{1/d} &\approx \log\log p. \end{aligned}$$

This leads to

$$v \approx \exp((\log p)^{1/3}(\log\log p)^{2/3}) = L_p(1/3, c)$$

for some $c$. To find this $c$, a bit more careful analysis is required. We find a good $c$ by choosing

$$d = (3^{1/3} + o(1))\left(\frac{\log p}{\log\log p}\right)^{1/3}.$$

Careful analysis will indeed show that this $d$ will give $c = (8/9)^{1/3}$, which means that the sieving step will run in

$$\begin{aligned} v^2 &= \exp(((64/9)^{1/3} + o(1))(\log p)^{1/3}(\log\log p)^{2/3}) \\ &= L_p(1/3, (64/9)^{1/3}). \end{aligned}$$

Note that the algorithm will run several times, once for each of the (large) prime factors $l$ of $p-1$. This will not affect the asymptotic run time as the number of prime factors of $p-1$ is $\log_2(p-1) = y^{o(1)}$ at maximum. Using

the Chinese remainder theorem in the end is trivial compared to the rest of the algorithm. Remember that, if the factorization of $p-1$ is not known, one can use the factorization version of the general number field sieve to find the factors in the same run time. Also, Wiedemann's algorithm [15] for solving sparse linear systems in $\mathbb{F}_p$ will make the linear algebra step possible to do in $y^{2+o(1)}$ steps, according to Buhler et. al. [2, page 84]. This means that the linear algebra step will not be slower than other steps asymptotically, giving us the above given bound as the actual bound for the whole algorithm.

# 4 An overview of algorithms

The preceding algorithm considers the discrete logarithm problem in the prime field $\mathbb{F}_p$. As we know, other finite fields exist, namely the prime power fields, $\mathbb{F}_{p^n}$. To classify the algorithms that solve the discrete logarithm in these fields, it is customary to divide the prime power algorithm into further cases, varying the characteristic and the extension. We do not cover algorithms for small, constant characteristic ($p = 2, 3, 5, 7$). This overview is based on the function field sieve presented by Joux and Lercier [7] and two number field sieve algorithms presented by Joux et. al. [8].

Algorithms for finding the discrete algorithm in $L_{p^n}(1/3, c)$ exist for all finite fields. Even so, the parameter $c$ changes for the different fields. The interesting question for medium prime fields that differentiates the fields is how fast $p$ grows with $p^n$. To denote this, we compare $p$ to $L_{p^n}(l_p, a)$ where $l_p$ varies. There are divides at $l_p = 1/3$ and $l_p = 2/3$.

If $p < L_{p^n}(1/3, a)$ the function field sieve gives $c = (\frac{32}{9})^{1/3}$. Note that this also includes the case when $p$ is fixed.

If $L_{p^n}(1/3, a) < p < L_{p^n}(2/3, a)$, a number field sieve algorithm gives $c = (\frac{128}{9})^{1/3}$.

If $p > L_{p^n}(2/3, a)$, another number field sieve algorithm gives $c = (\frac{64}{9})^{1/3}$. Note that $l_p \to \infty$ corresponds to the prime case, and that we found $c = (\frac{64}{9})^{1/3}$ for the prime case, like we should have.

At the dividing points, that is $l_p = 1/3$ and $l_p = 2/3$, the situation is a bit more complicated. Here, $c$ depends the other parameter in the growth of $p$, $a$. To get a better view on these dividing points, check Joux and Lercier [7, section 3] for the function field sieve and Joux et. al. [8, section 5] for the number field sieve.

# 5 The number field sieve in prime power fields

## 5.1 Overview

We now wish to consider the discrete logarithm problem in a prime power field, $\mathbb{F}_{p^n}$. The algorithm that is presented here will work for certain prime power fields, with "medium base prime". This simply means that $p$ is larger than the really small primes $(2, 3, 5...)$, and the extension degree is also larger than a small number like 2 or 3. We are in neither of those extreme cases. In the run time section we will give precise parameters on when the algorithm runs best. In the best case, the algorithm will run in $L_{p^n}(1/3, (64/9)^{1/3})$, like the prime algorithm. The disposition will follow Joux and Lercier [8] and the preceding algorithm closely.

Just as in the prime case, we want to find $\log_t g$, only now $t, g \in \mathbb{F}_{p^n}^*$. Our first hurdle is to find a representation of $\mathbb{F}_{p^n}$. In $\mathbb{F}_p$, this part is obvious, we simply took the integers modulo $p$. In $\mathbb{F}_{p^n}$, we start by taking polynomials with coefficients that are integers modulo $p$ ($\mathbb{F}_p[x]$), but we also find a irreducible polynomial $f_1$ of degree $n$, and divide out by this. So we will in other words use

$$\mathbb{F}_{p^n} \cong \mathbb{F}_p[x]/\langle f_1(x)\rangle$$

as our model for $\mathbb{F}_{p^n}$.

In fact, there are several polynomials which can be used as our $f_1$ here. We choose $f_1$ with as small coefficients as possible. After we have found $f_1$, we now define $f_2$ so they both have a common root $\mu$ in $\mathbb{F}_{p^n}$, as we did in the prime case. Several options are obviously possible here. We will use the simple one where we define $f_2 = f_1 + p$. Since we now get $f_1 \equiv f_2 \pmod{p}$, the polynomials will have a common root $\mu$ in $\mathbb{F}_{p^n}$, in fact all roots will be shared. We now use the two polynomials to define number fields. Use roots $\alpha_1, \alpha_2 \in \mathbb{C}$ of $f_1$ and $f_2$ to define the number fields $\mathbb{Q}[\alpha_1]$ and $\mathbb{Q}[\alpha_2]$, respectively. We now consider a factor $l$ of $p^n - 1$. Like we did in the prime algorithm, we will consider the problem modulo this $l$, and do this for all $l$, before we use the Chinese remainder theorem to return to the answer.

## 5.2 Sieving

In the corresponding section in the prime algorithm, we defined the rational and algebraic factor bases, $\mathcal{R}$ and $\mathcal{A}$. In this, we define two algebraic factor

bases for the number fields, $\mathcal{A}_1$ and $\mathcal{A}_2$. These consist of pairs $(q, r)$ representing first degree prime ideals in their respective number fields, as seen in (4). We let this consist of primes up to a smoothness bound, $y$, similarly to before. We sieve in the same way as before: choose $v$ and consider $(a, b)$ with $0 \leq b \leq v$ and $|a| \leq v$. We take the norms $N(a + b\alpha_1)$ and $N(a + b\alpha_2)$ of each element and divide them by the primes that lie under the prime ideals if the equation $a \equiv -br \pmod{q}$ is fulfilled.

The improvements involving logarithms and changing division to addition used in the prime algorithm also apply here.

## 5.3 Logarithmic maps

We have relations involving ideals in our number fields that we wish to transform into multiplicative relations involving elements. We can then take logarithms to get linear equations.

### 5.3.1 Trivial ideal class group and computable unit group

First we consider the case with trivial ideal class group and computable unit group. Trivial ideal class group implies that all ideals are principal.

From the sieving step we have relations of the form

$$\langle a + b\alpha \rangle = \prod_i \mathfrak{q}_i^{e_i}$$

where $\mathfrak{q}_i$ is a prime ideal in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. Since all ideals are principal we get that

$$a + b\alpha = u \prod_i \gamma_i^{e_i}$$

where $u$ is a unit in $\mathcal{O}_{\mathbb{Q}[\alpha]}$ and $\langle \gamma_i \rangle = \mathfrak{p}_i$. We now apply Dirichlet's unit theorem (theorem 9) to find the structure of the unit group $\mathcal{O}^*_{\mathbb{Q}[\alpha]}$,

$$\mathcal{O}^*_{\mathbb{Q}[\alpha]} \cong G \times \mathbb{Z}^{r_1 + r_2 - 1}$$

where $r_1$ is the number of real embeddings, $r_2$ is the number of conjugate pairs of complex embeddings and $G = \langle u_0 \rangle$ of order $m$. We use the notation $r = r_1 + r_2 - 1$. The $r + 1$ generators of the unit group is called fundamental units.

Assume we can compute fundamental units $u_0, u_1, \ldots, u_r$. We can then write $u = u_0^{n_0} u_1^{n_1} \ldots u_r^{n_r}$. We use this equation to define $r$ maps (indexed by $1 \leq i \leq r$)

$$\begin{aligned} \lambda_i : \mathcal{O}^*_{\mathbb{Q}[\alpha]} &\to \mathbb{Z} \\ u &\mapsto n_i. \end{aligned}$$

In addition we have a map

$$\begin{aligned} \lambda_0 : \mathcal{O}^*_{\mathbb{Q}[\alpha]} &\to \mathbb{Z}_m \\ u &\mapsto n_0. \end{aligned}$$

We see that all these maps are logarithmic. These maps now define a decomposition

$$a + b\alpha = \prod_{i=0}^{r} u_i^{\lambda_i(u)} \prod_i \gamma_i^{e_i}.$$

Taking logarithms on the above equation, we reach our goal

$$\log_t(a + b\alpha) \equiv \sum_{i=0}^{r} \lambda_i(u) \log_t u_i + \sum_i e_i \log_t \gamma_i \pmod{p^n - 1}.$$

Note that this equation is modulo $p^n - 1$, not modulo a large prime factor, $l$.

### 5.3.2 General number fields

We assume that the large prime factor $l$ of $p^n - 1$ does not divide $h$, the order of the ideal class group. This is a very minor restriction. From sieving we have relations on the form

$$\langle a + b\alpha \rangle = \prod_i \mathfrak{q}_i^{e_i}.$$

Since $\mathfrak{q}$ isn't a principal ideal in general, we need some procedure to make sure we are dealing with principal ideals. We do this by raising the relation to the $h$-th power. Given $\delta_i = \mathfrak{q}_i^h$, we know that $\delta_i$ is principal because raising the ideals to the $h$-th power will make the ideal class group trivial. We then have

$$\begin{aligned} \langle a + b\alpha \rangle^h &= \prod_i \mathfrak{q}_i^{e_i h} \\ \langle a + b\alpha \rangle^h &= \prod_i \delta_i^{e_i} \\ (a + b\alpha)^h &= u \prod_i \delta_i^{e_i} \end{aligned}$$

for a unit $u$.

Since we in general have no way to compute the unit group, it not a good strategy to take logarithms at this point, as $\log_t u$ will be a new unknown in each relation.

This problem will be circumvented by using the ideas in the section on Schrokauer maps in the prime algorithm. The big insight here is to not work over the whole unit group $\mathcal{O}^*_{\mathbb{Q}[\alpha]}$, but rather over $l$-th powers of units $\mathcal{O}^*_{\mathbb{Q}[\alpha]}/(\mathcal{O}^*_{\mathbb{Q}[\alpha]})^l$. This is possible because if $u \in (\mathcal{O}^*_{\mathbb{Q}[\alpha]})^l$, then $\log_t u \equiv 0 \pmod{l}$. We again assume that $l$ does not ramify in $\mathcal{O}_{\mathbb{Q}[\alpha]}$. We repeat some of the argument given in the prime algorithm to ease the exposition. Consider the set $\Gamma$ we defined,

$$\Gamma = \{\gamma \in \mathcal{O}_{\mathbb{Q}[\alpha]} | N(\gamma) \not\equiv 0 \pmod{l}\}.$$

Remember that all $a + b\alpha \in \Gamma$ since they are smooth, so we have $l \nmid N(a + b\alpha)$. Also note that $\mathcal{O}^*_{\mathbb{Q}[\alpha]} \subseteq \Gamma$, since the norm of a unit is one. Now, similarly to how we defined the maps in the prime algorithm, we write $\epsilon = l^D - 1$ with $D$ being the least common multiple of the irreducible factors of $f(x) \pmod{l}$. Then we get

$$\gamma^\epsilon \equiv 1 \pmod{l}.$$

We define a map

$$\begin{aligned} \lambda : \Gamma &\to l\mathcal{O}_{\mathbb{Q}[\alpha]}/l^2\mathcal{O}_{\mathbb{Q}[\alpha]} \\ \gamma &\mapsto (\gamma^\epsilon - 1) + l^2\mathcal{O}_{\mathbb{Q}[\alpha]} \end{aligned}$$

Using the module structure of $\mathcal{O}_{\mathbb{Q}[\alpha]}$, we fix a basis $\{b_i l + l^2\mathcal{O}_{\mathbb{Q}[\alpha]}\}_{i=1,\ldots,d}$ for $l\mathcal{O}_{\mathbb{Q}[\alpha]}/l^2\mathcal{O}_{\mathbb{Q}[\alpha]}$ and a basis $\{b_i l + l^2\mathcal{O}_{\mathbb{Q}[\alpha]}\}_{i=1,\ldots,r}$ for $\lambda(\mathcal{O}^*_{\mathbb{Q}[\alpha]})$. We again define the projections $\lambda_i : \Gamma \to \mathbb{Z}_l$ by

$$\gamma^\epsilon - 1 \equiv l \sum_{i=1}^{d} \lambda_i(\gamma) b_i \pmod{l^2}.$$

From theorem 13 we know that $\lambda$ is a logarithmic homomorphism. This means that $\lambda_i : \mathcal{O}^*_{\mathbb{Q}[\alpha]} \to \mathbb{Z}_l$ are homomorphisms for $i = 1, \ldots, n$.

We now work over $l$-th powers of units; $\mathcal{O}^*_{\mathbb{Q}[\alpha]}/(\mathcal{O}^*_{\mathbb{Q}[\alpha]})^l$. Consider $r$ of these maps together in a homomorphism;

$$\begin{aligned} \bar{\lambda} : \mathcal{O}^*_{\mathbb{Q}[\alpha]}/(\mathcal{O}^*_{\mathbb{Q}[\alpha]})^l &\to \mathbb{Z}_l^r \\ u &\mapsto (\lambda_1(u), \ldots, \lambda_r(u)). \end{aligned}$$

We now prove a theorem about $\bar{\lambda}$.

**Theorem 17.** Given that $\bar{\lambda}$ is injective, $u \in \mathcal{O}^*_{\mathbb{Q}[\alpha]}$ is an $l$th power if and only if $\bar{\lambda}(u) = 0$.

*Proof.* If $u$ is an $l$-th power, it is the identity in $\mathcal{O}^*_{\mathbb{Q}[\alpha]}/(\mathcal{O}^*_{\mathbb{Q}[\alpha]})^l$, which is sent to the identity in $\mathbb{Z}_l^r$, which is zero.

If $\bar{\lambda}(u) = 0$, we assume $u \neq a^l$ for any $a \in \mathcal{O}^*_{\mathbb{Q}[\alpha]}$, to get a contradiction. Since $\bar{\lambda}$ is injective, we get $u \neq a^l \Rightarrow \bar{\lambda}(u) \neq \bar{\lambda}(a^l)$. But $\bar{\lambda}(u) = 0 = \bar{\lambda}(a^l)$, a contradiction. □

To justify why we can assume $\bar{\lambda}$ to be injective, realize that if $\mathcal{O}^*_{\mathbb{Q}[\alpha]}$ does not contain the primitive $l$-th roots of unity, then $\mathcal{O}^*_{\mathbb{Q}[\alpha]}/(\mathcal{O}^*_{\mathbb{Q}[\alpha]})^l$ has $l^r$ elements, making $\bar{\lambda}$ an isomorphism. We will make this assumption. Now there exist units $u_1, \ldots, u_r \in \mathcal{O}^*_{\mathbb{Q}[\alpha]}$ with the property that $\lambda_i(u_i) = 1$ and $\lambda_i(u_j) = 0$ whenever $i \neq j$.

We are now close to how we solved this in the easy case above, since we can write any unit $u \in \mathcal{O}_{\mathbb{Q}[\alpha]}$ as

$$u = \xi^l \prod_{i=1}^{r} u_i^{\lambda_i(u)}$$

for some unit $\xi$. We multiply each $\delta_i$ with a unit,

$$\delta_i' = \delta_i \prod_{j=1}^{r} u_j^{-\lambda_j(\delta_i)}.$$

Multiplying an ideal by a unit does not change the ideal, so $\delta_i'$ still generates $\mathfrak{q}_i^h$. This definition makes $\lambda_j(\delta_i') = 0$ for $j$ up to $r$. We now have

$$(a + b\alpha)^h = u' \prod_i (\delta_i')^{e_i}$$

for a unit $u'$. Applying the unit equation to $u'$, we get

$$(a + b\alpha)^h = \xi_{a,b}^l \prod_{i=1}^{r} u_i^{\lambda_i(u')} \cdot \prod_i (\delta_i')^{e_i}.$$

Since $\lambda_j(\delta_i') = 0$ for all $j$ up to $r$, we get that $\lambda_j(u') = h\lambda_j(a + b\alpha)$, which gives

$$(a + b\alpha)^h = \xi_{a,b}^l \prod_{i=1}^{r} u_i^{h\lambda_i(a+b\alpha)} \cdot \prod_i (\delta_i')^{e_i}.$$

We now take logarithms modulo $l$ to get

$$\log_t(a+b\alpha) \equiv \sum_{i=1}^{r} \lambda_i(a+b\alpha)\log_t u_i + \sum_i e_i h^{-1}\log_t \delta_i' \pmod{l}. \quad (9)$$

We see that $\xi_{a,b}$ disappeared modulo $l$, like we saw in the calculating logarithms section in the prime algorithm.

## 5.4 Linear Algebra

We will cover the linear algebra step assuming we have general number fields (not trivial ideal class group, nor computable unit group). The goal is to find logarithms of all smooth elements $a+b\alpha$.

We now consider the above relation (9) in both number fields, $\mathbb{Q}[\alpha_1]$ and $\mathbb{Q}[\alpha_2]$. Note that the numbers and elements in the relation will be different for the different number fields, even if we use the same notation for both of them. Remember that $\langle\delta_i'\rangle = \langle a+b\alpha\rangle^h$ in (9). Thus, we can consider $h^{-1}\log_t \delta_i'$ to represent the logarithm of the $i$-th ideal $\langle a+b\alpha\rangle$. This is called the virtual logarithm of the ideal. Applying (9) to both our number fields gives us relations involving up to $2n\pi(y)$ smooth numbers and $2r$ units, where $\pi(y)$ is the number of smooth integers. This means that it is necessary to collect $2n\pi(y)+2r \approx 2n\pi(y)$ relations in the sieving stage. Given enough relations, we solve the resulting linear system to find the virtual logarithm $\log_t a+b\alpha$ for all smooth numbers $a+b\alpha$, in both number fields. Note that we do this step several times (once for each $l$), and then use the Chinese remainder theorem like we did in the prime case.

Note that $g$ does not affect the linear system here, unlike in the prime algorithm. The advantage of this is that one can find several discrete logarithms in base $t$ without redoing the linear algebra and sieving.

## 5.5 Finding the logarithm

The challenge now is to use the calculated virtual logarithms to find the logarithm we want. We want to find the discrete logarithm $\log_t g$ in $\mathbb{F}_{p^n}$. This is done using a technique called "special-$\mathfrak{q}$" descent.

First we search for an element $z = g^i x^j$ in $\mathbb{F}_{p^n} \cong \mathbb{F}_p[x]/\langle f_1(x)\rangle$, where $i,j \in \mathbb{N}$. We want $z$ to satisfy two properties. The first is that we want $\langle z\rangle \in \mathbb{Q}[\alpha_1]$ to have a norm that have "smallish" prime factors. More

precisely we want the prime factors of $N(\langle z \rangle)$ to be smaller than some bound $B_1 \in L_{p^n}(2/3, 1/3^{1/3})$. The second property is that we want $N(\langle z \rangle)$ to be squarefree, which means that it factors completely into first degree prime ideals.

It seems that if we can find such a $z$, then we can use the logarithms we calculated in the linear algebra step to get the answer. This is not possible yet, since the factors of $N(\langle z \rangle)$ may be larger than the elements in the smoothness bases. That is, we probably have $v < B_1$. The squarefree condition makes sure that all considered prime ideals are of degree one, which all ideals in the smoothness bases are.

We assume that $z$ has the same probability of having squarefree factorization into small primes as a random number of the same size. According to Joux et. al. [8] there is a statement in Ivić and Tenenbaum [4], that says that we can write this as the product of the probability of the two properties. The squarefree condition quickly tends to $6/\pi^2$, which vanishes into the $o(1)$ in the $L$-notation. The normal probability of a number having small prime factors will be used and analyzed in the section where we determine the run time.

After we have found a $z$ fulfilling our criteria, we factor the principal ideal generated by $z$ into degree one prime ideals of small norm. We note that there will be prime ideals in the factorization not contained in the factor bases $\mathcal{A}_1$ and $\mathcal{A}_2$, since their norm was allowed to be bigger than the smoothness bound $v$. These will be dealt with by using the special-$\mathfrak{q}$ descent.

### 5.5.1 Special-$\mathfrak{q}$ descent

We wish to find the logarithm of a prime ideal $\mathfrak{q}$ that is too big to lie in our factor base. We start by sieving on pairs $(a, b)$, chosen in such a way that $\mathfrak{q}$ divides $a+b\alpha_1$ in $\mathbb{Z}[\alpha_1]$. When we find a pair $(a, b)$ such that $N(a+b\alpha_1)/N(\mathfrak{q})$ and $N(a+b\alpha_2)$ factor into primes smaller than $B_2 < B_1$, we iterate the descent. When we iterate, we consider special-$\mathfrak{q}$ ideals in both number fields, and we get a lower bound on the primes for each iteration. We continue until the bound is lower than $v$, when we know all logarithms. We use these to backtrack to find the logarithms of each special-$\mathfrak{q}$ ideal, and then the logarithm of $z$.

### 5.5.2 The last step

To find $\log_t g$ we repeat the above procedure twice, generating $z = g^i x^j$ and $z' = g^{i'} x^{j'}$ that we know the logarithms of. When this is done, we make sure that $ij' \neq i'j \pmod{p^n - 1}$, so that it is possible to generate a logarithm. We will show that this will give us $\log_t g$. We have two sets of equations we will use,

$$\begin{aligned} z &= g^i x^j, \qquad \log_t z = r \\ z' &= g^{i'} x^{j'}, \qquad \log_t z' = r' \end{aligned}$$

where $i, j, r \in \mathbb{N}$ and the rest of the letters are group elements. If we put these two together, we get the following equations

$$g^i x^j = t^r, \qquad g^{i'} x^{j'} = t^{r'}.$$

The second equation gives us $x^{j'} = t^{r'} g^{-i'}$. Raise the first equation to the power of $j'$ to get

$$\begin{aligned} g^{ij'} x^{jj'} &= t^{rj'} \\ g^{ij'} (x^{j'})^j &= t^{rj'} \\ g^{ij'} (t^{r'} g^{-i'})^j &= t^{rj'}. \end{aligned}$$

Solving the last equation is straight forward, and gives us

$$\log_t g = \frac{rj' - r'j}{ij' - i'j} \pmod{p^n - 1},$$

solving the discrete logarithm problem.

## 5.6 Run time

In this section we want to show that the run time of the prime power algorithm is the same as the prime algorithm, for a certain relationship between $p$ and $n$. This run time is $L_{p^n}(1/3, (64/9)^{1/3})$. We will use some of the results in that section to ease the exposition here.

First we assume that $n$ and $p$ fulfill the following relations

$$n = \frac{1}{c} \Big( \frac{\log p^n}{\log \log p^n} \Big)^{1/3}$$

$$p = \exp\left(c(\log p^n)^{2/3}(\log\log p^n)^{1/3}\right)$$

for some constant $c$. This puts us in one of the dividing points in section 4, namely $l_p = 1/3$. Note that these definitions are compatible with any $c \neq 0$, since we have

$$\begin{aligned}
&\exp\left(c((\log p^n)^{2/3}(\log\log p^n)^{1/3}\right)^{\frac{1}{c}\left(\frac{\log p^n}{\log\log p^n}\right)^{1/3}} \\
=\ &\exp\left(c((\log p^n)^{2/3}(\log\log p^n)^{1/3}\frac{1}{c}\left(\frac{\log p^n}{\log\log p^n}\right)^{1/3}\right) \\
=\ &\exp\left((\log p^n)^{2/3}(\log p^n)^{1/3}\right) = p^n.
\end{aligned}$$

Like we did in the prime algorithm, we will make the sieve limit $v$ and smoothness bound $y$ equal. We will assume that these will be defined as

$$v = y = \exp\left(c'(\log p^n)^{1/3}(\log\log p^n)^{2/3}\right)$$

for some constant $c'$, just like what happened in the prime case algorithm.

Remember that $f_1$ has small coefficients. Let $b_0$ be the bound on the coefficients. To find the norm, we remember that we sieve over $v$ values and since $N(a + b\alpha_1) = (-b)^n f(-\frac{a}{b})$ we get

$$N(a + b\alpha_1) \leq (n+1)b_0 v^n = y^{n+o(1)}.$$

Similarly, the bound on the norm of $a + b\alpha_2$ will be $py^{n+o(1)}$. We consider the bound on the product of the norms, which will be $py^{2n+o(1)}$. Using the expressions we assume about $p$, $n$ and $y$ we get

$$\begin{aligned}
&N(a + b\alpha_1)N(a + b\alpha_2) \leq py^{2n+o(1)} \\
&= \exp\left(c(\log p^n)^{2/3}(\log\log p^n)^{1/3}\right) \\
&\cdot \exp\left(c'(\log p^n)^{1/3}(\log\log p^n)^{2/3}\frac{2}{c}\left(\frac{\log p^n}{\log\log p^n}\right)^{1/3} + o(1)\right) \\
&= \exp\left(c(\log p^n)^{2/3}(\log\log p^n)^{1/3}\right)\exp\left(\frac{2c'}{c}(\log p^n)^{2/3}(\log\log p^n)^{1/3} + o(1)\right) \\
&= \exp\left(\left(c + \frac{2c'}{c} + o(1)\right)(\log p^n)^{2/3}(\log\log p^n)^{1/3}\right) \\
&= L_{p^n}(2/3, c + 2c'/c).
\end{aligned}$$

According to Joux et.al. [8], there is a theorem in Canfield et. al. [3] that defines the probability of a random number being smooth. Specifically, a random number less than $L_{p^n}(r, \gamma)$ is $L_{p^n}(s, \delta)$-smooth with probability $L_{p^n}(r - s, -\gamma(r - s)/\delta)$. Remember that we are not only looking at smooth numbers, but smooth numbers with squarefree factorizations. Like discussed earlier, we can hide the factor $6/\pi^2$ from the squarefree condition in the $o(1)$ part of the $L$-notation. We use the heuristic we used in the prime algorithm, and assume that our number behave like random numbers.

We realize that $y = L_{p^n}(1/3, c')$. So the probability of $N(a + b\alpha_1)N(a + b\alpha_2) \leq L_{p^n}(2/3, c + 2c'/c)$ being $y$-smooth is

$$\begin{aligned} & L_{p^n}(2/3 - 1/3, -(c + 2c'/c)(2/3 - 1/3)/c') \\ = \; & L_{p^n}(1/3, -(1/3)(c/c' + 2/c)). \end{aligned}$$

To minimize the total runtime, we balance the sieving step and the linear algebra step, like in the prime algorithm. This is the same as balancing the rows and columns, as was done in the prime algorithm. In other words, we need to balance $v = L_{p^n}(1/3, c')$ and the average time it will take to find a $y$-smooth number. This average time will be the inverse of the probability of the sieved numbers being smooth. Explicitly, the following equation corresponds to this balancing,

$$L_{p^n}(1/3, c') = L_{p^n}(1/3, (1/3)(c/c' + 2/c)),$$

which simplifies to

$$c' = 1/3(c/c' + 2/c).$$

We solve this equation for $c'$,

$$\begin{aligned} 0 &= c'^2 - \frac{2}{3c}c' - \frac{c}{3} \\ c' &= \frac{1}{3c} \pm \sqrt{\frac{1}{9c^2} + \frac{c}{3}} \\ &= \frac{1}{3}\Big(\frac{1}{c} \pm \sqrt{3c + \frac{1}{c^2}}\Big). \end{aligned}$$

Here, there is a question of whether we should use the positive or negative square root. We use the positive square root in the following, but we note that we would get the same result if we chose the negative square root.

Like in the prime algorithm, the step that takes the most time will be the sieving, which will run in $v^2$ steps. This means that the complexity of the algorithm will be $L_{p^n}(1/3, 2c')$. We wish to know when this is minimal. We do this by differentiation,

$$3\frac{\mathrm{d}c'}{\mathrm{d}c} = \frac{3 - 2c^{-3}}{2\sqrt{3c + c^{-2}}} - \frac{1}{c^2} = \frac{3c^3 - 2 - 2\sqrt{3c^3 + 1}}{2c^2\sqrt{3c^3 + 1}}.$$

We continue only with the nominator equal to zero,

$$3c^3 - 2 - 2\sqrt{3c^3 + 1} = 0.$$

By choosing $u = 3c^3$, we can solve this

$$\begin{aligned} u - 2 - 2\sqrt{u + 1} &= 0 \\ (u - 2)^2 &= 4(u + 1) \\ u^2 - 8u &= 0 \\ u &= 8 \\ c &= 2 \cdot (1/3)^{1/3}, \end{aligned}$$

where $u = 0$ could not have been a solution, as it had made the denominator above equal to zero. We see that $c = 2 \cdot (1/3)^{1/3}$ corresponds to $c' = 2 \cdot (1/3)^{2/3}$. It is clear that this is not a maximum because i.e. $c = 2$ corresponds to $c' = 3/2$. This $c$ gives us the run time of $L_{p^n}(1/3, 2c') = L_{p^n}(1/3, (64/9)^{1/3})$, which is the same as the prime algorithm.

We note that the algorithm described here is one of many in a class of algorithms that works for different $c$. These are given in Joux et. al. [8], and are included in section 4 in this text.

# 6 Concluding remarks

We have given a description of the general number field sieve for discrete logarithms in prime fields and in prime power fields. Both algorithms considered have the same run time, $L_Q(1/3, (64/9)^{1/3})$, where $Q$ is the size of the finite field. The family of algorithms described in section 4 contains the best known algorithms for solving the discrete logarithm when the size of the finite field becomes large enough, i.e. they have the fastest asymptotic run time known. Interestingly, they are also practically applicable. The practicality of these algorithms makes them interesting to study in the cryptographic context.

A lot of work goes into trying to improve the second parameter of the $L$-notation. Improvements here gives faster algorithms, but no large breakthrough will come as a result of improving the second parameter. A breakthrough will come with an improvement in the first parameter, ideally to zero, for a polynomial algorithm. If such a breakthrough happens, it is fully possible that discrete logarithms will continue to be used, but the underlying group will not be a finite field, but rather something like an elliptic curve.

# References


[1] Robert B. Ash. *A course in algebraic number theory*. Courier Corporation, 2010.

[2] Joe P. Buhler, Hendrik W. Lenstra Jr, and Carl Pomerance. Factoring integers with the number field sieve. In *The development of the number field sieve*, pages 50–94. Springer, 1993.

[3] E. Rodney Canfield, Paul Erdös, and Carl Pomerance. On a problem of Oppenheim concerning "factorisatio numerorum". *Journal of Number Theory*, 17(1):1–28, 1983.

[4] Aleksandar Ivić and Gérald Tenenbaum. Local densities over integers free of large prime factors. *The Quarterly Journal of Mathematics*, 37(4):401–417, 1986.

[5] Christina Jamroz. Ideal class group. Univeristy of Notre Dame, 2009.

[6] Antoine Joux and Reynald Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the gaussian integer method. *Math. Comput.*, 72(242):953–967, 2003.

[7] Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 254–270. Springer, 2006.

[8] Antoine Joux, Reynald Lercier, Nigel P. Smart, and Frederik Vercauteren. The number field sieve in the medium prime case. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 326–344. Springer, 2006.

[9] Daniel A. Marcus. *Number fields*, volume 1995. Springer, 1977.

[10] Per Kristian Ørke. Improving on the number field sieve. Master's thesis, NTNU, 2015.

[11] Oliver Schirokauer. Discrete logarithms and local units. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 345(1676):409–423, 1993.

[12] Oliver Schirokauer. The impact of the number field sieve on the discrete logarithm problem in finite fields. In *Proceedings of the 2002 Algorithmic Number Theory workshop at MSRI*, 2008.

[13] Ruben Grønning Spaans. Number field sieve. Master's thesis, NTNU, 2013.

[14] Elin Margrete Trondsen. The number field sieve. Master's thesis, NTNU, 2012.

[15] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, Jan 1986.