Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS

Anna Godušová

Number Field Sieve for Discrete Logarithm

Department of Algebra

Supervisor of the master thesis:	RNDr. Přemysl Jedlička, Ph.D.
Study programme:	Mathematics
Study branch:	Mathematical Methods of Information Security

Prague 2015

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague date 4.12.2015 Anna Godušová

Název práce: Síto v číselném tělese pro diskrétní logaritmus

Autor: Anna Godušová

Katedra: Katedra algebry

Vedoucí diplomové práce: RNDr. Přemysl Jedlička, Ph.D., TF ČZÚ

Abstrakt: Mnoho dnešních kryptografických systémů, jako například protokol Diffie-Hellman, je založených na problému diskrétního logaritmu. Síto v číselném tělese je algoritmus řešící faktorizaci velkých celých čísel, nové poznatky ale ukazují, že může být použit i na problém diskrétního logaritmu.

V této práci studujeme síto v číselném tělese pro diskrétní logaritmus a porovnáváme ho se sítem v číselném tělese pro faktorizaci. Oba algoritmy jsou založeny na stejném principu, ale v jednotlivých krocích nalézáme velké rozdíly.

Klíčová slova: Síto v číselném tělese, diskrétní logaritmus, NFS

Title: Number Field Sieve for Discrete Logarithm

Author: Anna Godušová

Department: Department of Algebra

Supervisor: RNDr. Přemysl Jedlička, Ph.D., TF ČZÚ

Abstract: Many of today's cryptographic systems are based on the discrete logarithm problem, e.g. the Diffie-Hellman protocol. The number field sieve algorithm (`NFS`) is the algorithm solving the problem of factorization of integers, but latest works show, it can be also applied to the discrete logarithm problem.
In this work, we study the number field sieve algorithm for discrete logarithm and we also compare the `NFS` for discrete logarithm with the `NFS` for factorization. Even though these `NFS` algorithms are based on the same principle, many differences are found.

Keywords: Number field sieve, discrete logarithm, NFS

I would like to thank to my supervisor RNDr. Přemysl Jedlička, Ph.D. for his guidance and helpful advices.

I would like to thank to Monika Steinová for reviewing some of the chapters.

I am thankful to my fiancé Marek Paška, for all the support, trust and hours spent of reading this thesis.

Contents

Introduction

Many public key cryptographic systems are based on the discrete logarithm problem as it is considered to be difficult to find x with the knowledge of t, u and p, where $t^x \equiv u \pmod{p}$. This problem has been widely studied and a lot of different approaches have been taken. Nowadays, the asymptotically fastest algorithms for solving the discrete logarithm are the number field sieve algorithm, first introduced by Gordon in [1], and the function field sieve algorithm, first introduced by Adleman in [2]. These algorithms are based on the Index Calculus method introduced in 1979 in [3] and then analyzed in detail by Odlyzko in [4]. The Index Calculus algorithm is based on the principle that the discrete logarithms of small elements of a factor base are calculated first and then the searched discrete logarithm is computed with respect to these precomputed discrete logarithms. In the case of number and function field sieve algorithms, two kinds of groups are often considered: elliptic curves, and multiplicative groups of finite fields in the form $\mathbb{F}_{p^n}$. The multiplicative groups of finite fields are often also subdivided and studied separately according to the size of p and n. For example, if $n = 1$, then the best known algorithm for $\mathbb{F}_{p^1}$ is the number field sieve. For a fixed characteristic and n tending to infinity, the best algorithm is the function field sieve.

This work analyzes the number field sieve algorithm for discrete logarithm for multiplicative groups of the finite fields in the form $\mathbb{F}_{p^n}$, where $n > 1$ and p being a medium to large prime. It also describes the number field sieve algorithm for factorization and compares the differences of the number field sieve for factorization and for discrete logarithm.

Chapter 2 covers the theory used in the number field sieve algorithm for discrete logarithm. The goal is not to mention all the basic definitions and theorems, but to present the important parts of the theory. We assume that the reader has the knowledge of the linear algebra, the number theory, properties of the finitely generated Abelian group and the theory of the finite fields. In this chapter, we cover number fields, Dedekind domains, units in the number fields, and introduce theorems for the ideal factorization in extensions.

In Chapter 3, we describe the Index Calculus algorithm which gives the relevant background for computing the number field sieve algorithm. We illustrate the algorithm on an example using integers followed by a description of the algorithm for polynomials.

Chapter 4 analyzes the number field sieve algorithm for discrete logarithm in detail. We focus mainly on the parts that differ from the number field sieve algorithm for factorization, but we also include the description of the other components. Because it is a complex algorithm, we also provide two examples, one for $\mathbb{F}_p$ and one for $\mathbb{F}_{p^n}$, as an illustration.

Chapter 5 covers brief description of the number field sieve algorithm for factorization with focus on the problems of finding square powers and square roots.

Finally, in Chapter 6, we analyze the differences between the number field sieve algorithm for factorization and for discrete logarithm. Even though the algorithms are based on the same principle, many parts of the algorithms are different.

Chapter 1

Theory

In this chapter, we give the necessary background on the number fields, ring of integers, and Dedekind domains. We also cover the units in number fields and the theory for the factorization of ideals to prime ideals. The aim here is not to cover all the basic theory, but focus on the important definitions and theorems.

1.1 Number Fields

Definition 1.1. *Let $\alpha \in \mathbb{C}$. Then α is called an **algebraic number** if there exists $f \in \mathbb{Z}[X]$ such that $f(\alpha) = 0$ and f is not identically zero.*

Definition 1.2. *A **number field** K is a subfield of complex numbers having a finite degree over $\mathbb{Q}$.*

A number field is an algebraic extension of $\mathbb{Q}$ of a finite degree if and only if $K = \mathbb{Q}[\alpha]$, for some algebraic number $\alpha \in \mathbb{C}$. If α is a root of an irreducible polynomial over $\mathbb{Q}$ with degree n, then

$$K = \mathbb{Q}[\alpha] = \{a_0 + a_1\alpha + \ldots + a_{n-1}\alpha^{n-1} : a_i \in \mathbb{Q}, i = 0, 1, \ldots, n-1\}. \tag{1.1}$$

The set $\{1, \alpha, \ldots, \alpha^{n-1}\}$ is a basis of $\mathbb{Q}[\alpha]$ as a vector space over $\mathbb{Q}$.

Definition 1.3. *A complex number ϑ is an **algebraic integer** if it is a root of a monic polynomial with coefficients in $\mathbb{Z}$. Denote by $\mathbb{A}$ the set of all algebraic integers.*

Theorem 1.4. *Let ϑ be an algebraic integer and let f be its minimal polynomial. Then f is irreducible over $\mathbb{Q}$.*

We need the following lemma in later proofs.

Lemma 1.5. *Let R be a commutative ring and O, a finitely generated faithful R-module generated by $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Then, for an arbitrary nonzero element ρ such that $\rho\alpha_i \in O$, for all $i = 1, \ldots, n$, there exists a monic polynomial over R with a root ρ.*

Proof. Expressing each $\rho\alpha_i$ as an R-linear combination of generators, we obtain

$$\begin{pmatrix} \rho\alpha_1 \\ \vdots \\ \rho\alpha_n \end{pmatrix} = M \cdot \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix},$$

where $M \in R^{n\times n}$. Equivalently,

$$(\rho I - M)\begin{pmatrix}\alpha_1\\ \vdots\\ \alpha_n\end{pmatrix} = \begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}.$$

Which means that $\det(\rho I - M) = 0$. Therefore ρ is a root of a polynomial over R. □

We now describe some properties of the algebraic integers.

Lemma 1.6. *If ϑ and μ are algebraic integers then $\vartheta+\mu$ and $\vartheta\mu$ are also algebraic integers.*

Proof. If we find monic polynomials over $\mathbb{Z}$ with roots $\vartheta+\mu$ and $\vartheta\mu$ then, according to Definition 1.3, $\vartheta+\mu$ and $\vartheta\mu$ are algebraic integers. Let d_1 and d_2 be the degrees of monic polynomials over $\mathbb{Z}$ with ϑ and μ, respectively, as a root. Then the $\mathbb{Z}$-module $\mathbb{Z}[\vartheta,\mu]$ is generated by the set $M = \{\vartheta^i\mu^j, 0 \le i < d_1, 0 \le j < d_2\}$. Now, consider $\rho \in \{\vartheta+\mu, \vartheta\mu\}$. Following the previous lemma, there exists a polynomial over $\mathbb{Z}$ with a root ρ. □

The corollary of the previous lemma is the next important theorem.

Theorem 1.7. *The set $\mathbb{A}$ of algebraic integers in $\mathbb{C}$ is a ring.*

Now we can define an analogy of the integers in the number field K.

Definition 1.8. *The set of algebraic integers (ring of integers) of the number field K is the ring $\mathbb{A} \cap K$. We denote it by $\mathcal{O}_K$.*

Definition 1.9. *Let K be a number field, $n = [K : Q]$ and denote by $\sigma_1, \ldots, \sigma_n$ the distinct embeddings of K in $\mathbb{C}$, with $\sigma_i \mid \mathbb{Q} = id_Q$. Then we define two functions, the trace $T : K \to \mathbb{C}$ and the norm $N : K \to \mathbb{C}$ as*

$$T(\alpha) = \sum_{i=1}^{n} \sigma_i(\alpha), \text{ and}$$
$$N(\alpha) = \prod_{i=1}^{n} \sigma_i(\alpha).$$

The norm and the trace are sometimes denoted by $N_{\mathbb{Q}}^K$ and $T_{\mathbb{Q}}^K$ if we need to emphasize the fields.

Immediately from the definition we can see that $T(\alpha+\beta) = T(\alpha)+T(\beta)$ and $N(\alpha\beta) = N(\alpha)N(\beta)$ for all $\alpha, \beta \in K$. Moreover, for $a \in \mathbb{Q}$ we have $T(a) = na$ and $N(a) = a^n$.

Lemma 1.10. *Let K be a number field and $\mathcal{O}_K$ its ring of integers, $n = [K : \mathbb{Q}]$. Then, for $\alpha \in K$, $T(\alpha) \in \mathbb{Q}$ and $N(\alpha) \in \mathbb{Q}$ and for $\alpha \in \mathcal{O}_K$, $T(\alpha) \in \mathbb{Z}$ and $N(\alpha) \in \mathbb{Z}$.*

Proof. Let us choose some $\alpha \in K$ and let $d = [\mathbb{Q}[\alpha] : \mathbb{Q}]$. Then $[K : \mathbb{Q}[\alpha]] = \dfrac{n}{d}$ and only d embeddings of K in $\mathbb{C}$ are distinct over $\mathbb{Q}[\alpha]$. The $T_{\mathbb{Q}}^{\mathbb{Q}[\alpha]}(\alpha)$ is the second coefficient of the monic irreducible polynomial of α over $\mathbb{Q}$, and $N_{\mathbb{Q}}^{\mathbb{Q}[\alpha]}(\alpha)$ is the constant term. These coefficients are rational. We get $T_{\mathbb{Q}}^{K}(\alpha) = \frac{n}{d} T_{\mathbb{Q}}^{\mathbb{Q}[\alpha]}(\alpha)$, similarly for the norm. Since the monic polynomials of the algebraic integers have the coefficients over $\mathbb{Z}$, the norm and the trace are integers. □

Lemma 1.11. *Let K be a number field with its ring of integers $\mathcal{O}_K$ and let $\vartheta \in \mathcal{O}_K$. Then*

(i) ϑ is a unit in $\mathcal{O}_K$ if and only if $N(\vartheta) = \pm 1$;

(ii) if $N(\vartheta) = \pm p$, p a prime number, then ϑ is irreducible in $\mathcal{O}_K$.

Proof. (i) We have $N(1) = 1$. If $\vartheta\mu = 1$, then $1 = N(1) = N(\vartheta\mu) = N(\vartheta)N(\mu)$. Since $N(\vartheta), N(\mu) \in \mathbb{Z}$, then $N(\vartheta) = \pm 1$.
Let $n = [K : Q]$, $\sigma_1, \dots, \sigma_n$ be all distinct embeddings of K in $\mathbb{C}$ and let $\sigma_1 = \mathrm{id}_K$. Then $1 = N(\vartheta) = \prod_{i=1}^{n} \sigma_i(\vartheta) = \vartheta \prod_{i=2}^{n} \sigma_i(\vartheta) = \vartheta \cdot \mu$, where μ belongs to $\mathcal{O}_K$.

(ii) If $\vartheta = \mu_1 \cdot \mu_2$ then $N(\vartheta) = N(\mu_1)N(\mu_2)$ in $\mathbb{Z}$. Since $N(\vartheta)$ is a prime number, then either $N(\mu_1)$ or $N(\mu_2)$ has to be ± 1 and therefore a unit in $\mathcal{O}_K$ which follows from point (i). □

Another important role in the number theory is the discriminant.

Definition 1.12. *Let K and L be the number fields, $K \subseteq L$, and $n = [L : K]$. Let $\sigma_1, \dots \sigma_n$ denote the n embeddings of L in $\mathbb{C}$ such that the restrictions $\sigma_i | K = id_K$. For any n-tuple of elements $\alpha_1, \dots, \alpha_n \in L$, define the **discriminant** of $\alpha_1, \dots, \alpha_n$ to be*

$$disc(\alpha_1, \dots, \alpha_n) = |(\sigma_i(\alpha_j))_{i,j=1}^{n}|^2,$$

i.e., the square of the determinant of the matrix having $\sigma_i(\alpha_j)$ in the i^{th} row and j^{th} column.

We can express the discriminant in terms of the trace.

Lemma 1.13. *Let K and L be the number fields, $K \subseteq L$, and $n = [L : K]$. Then*

$$disc(\alpha_1, \dots, \alpha_n) = |(T_K^L(\alpha_i\alpha_j))_{i,j=1}^{n}|$$

for $\alpha_1, \dots, \alpha_n \in L$.

Proof. We have $(\det(M))^2 = \det(MM) = \det(M^T M)$ for any matrix M. We need to prove that the multiplication of i^{th} row and j^{th} column of the matrix $(\sigma_i(\alpha_j))_{i,j=1}^{n}$ gives the element on the position (i,j) of the matrix $(T_K^L(\alpha_i\alpha_j))_{i,j=1}^{n}$. Because

$$(\sigma_1(\alpha_i), \dots, \sigma_n(\alpha_i)) \cdot (\sigma_1(\alpha_j), \dots, \sigma_n(\alpha_j))^T =$$
$$= \sigma_1(\alpha_i)\sigma_1(\alpha_j) + \dots + \sigma_n(\alpha_i)\sigma_n(\alpha_j) = \sigma_1(\alpha_i\alpha_j) + \dots + \sigma_n(\alpha_i\alpha_j) = T_K^L(\alpha_i\alpha_j),$$

we prove the lemma. □

Corollary 1.14. *Let K be the number field,$n = [K : \mathbb{Q}]$ and $\alpha_1, \dots, \alpha_n \in K$. Then $disc(\alpha_1, \dots, \alpha_n) \in \mathbb{Q}$. If $\alpha_1, \dots, \alpha_n \in \mathcal{O}_K$ then $disc(\alpha_1, \dots, \alpha_n) \in \mathbb{Z}$.*

In the next section, we show how to determine if a set of the elements of the number field K is linearly independent and we define the connection between two bases using the discriminant. We also describe the structure of the ring of integers.

Theorem 1.15. *Let K be a number field, $n = [K : \mathbb{Q}]$. Then disc$(\alpha_1, \ldots, \alpha_n) = 0$ if and only if $\alpha_1, \ldots, \alpha_n \in K$ are linearly dependent over $\mathbb{Q}$.*

Proof. If $\alpha_1, \ldots, \alpha_n \in K$ are linearly dependent over $\mathbb{Q}$, then there exists $a_i \in \mathbb{Q}$ (not all zero) such that $\sum_{i=1}^{n} a_i\alpha_i = 0$. Let $\sigma_1, \ldots, \sigma_n$ be all distinct embeddings of K in $\mathbb{C}$ such that restrictions $\sigma_i|\mathbb{Q} = \mathrm{id}_{\mathbb{Q}}$. Then $\sum_{i=1}^{n} a_i\sigma_j(\alpha_i) = 0$ for $j = 1, \ldots, n$. Moreover, rows of the matrix $(\sigma_i(\alpha_j))_{i,j=1}^{n}$ are linearly dependent. We can again use the coefficients a_i and get $\mathrm{disc}(\alpha_1, \ldots, \alpha_n) = 0$.

Conversely, let $\alpha_1, \ldots, \alpha_n \in K$ be linearly independent over $\mathbb{Q}$. If $\mathrm{disc}(\alpha_1, \ldots, \alpha_n) = 0$, then the rows χ_i of the matrix $(T(\alpha_i\alpha_j))_{i,j=1}^{n}$ are linearly dependent and we can find the rational numbers $a_i, \ldots, a_n$ (not all 0) such that $a_1\chi_1 + \ldots + a_n\chi_n$ is the zero vector. From the linearly independence of the $\alpha_1, \ldots, \alpha_n$ we obtain $\alpha = a_1\alpha_1 + \ldots + a_n\alpha_n \neq 0$. Moreover, by considering only the j^{th} coordinate of each row, we obtain the fact that $T(\alpha\alpha_j) = \sum_{i=1}^{n} a_iT(\alpha_i\alpha_j) = 0$ for $j = 1, \ldots, n$. Since $\alpha_1, \ldots, \alpha_n$ are assumed to be linearly independent over $\mathbb{Q}$, they form a basis of K over $\mathbb{Q}$; it then follows that, also $\alpha\alpha_1, \ldots, \alpha\alpha_n$ forms a basis. But then $T(\beta) = 0$ for every $\beta \in K$ which is a contradiction since, for example, $T(1) = n$. □

Theorem 1.15 shows that every basis of K over $\mathbb{Q}$ has a nonzero discriminant. The next theorem describes the additive structure of the ring of integers. Proof of this theorem can be found in [5], Chapter 2.

Theorem 1.16. *Let K be a number field with its ring of integers $\mathcal{O}_K$ and let $n = [K : \mathbb{Q}]$. Then there exist $\vartheta_1, \ldots \vartheta_n \in \mathcal{O}_K$ such that $\mathcal{O}_K = \{\sum_{i=1}^{n} z_i \cdot \vartheta_i; z_i \in \mathbb{Z}\}$, i.e., $\mathcal{O}_K$ is a free abelian group of rank n.*

Definition 1.17. *Set $\{\vartheta_1, \ldots \vartheta_n\}$ is called an* ***integral basis*** *of $\mathcal{O}_K$, where $\mathcal{O}_K = \{\sum_{i=1}^{n} z_i \cdot \vartheta_i; z_i \in \mathbb{Z}\}$.*

A ring of integers has more than one integral basis. However all of them have the same discriminant. Thus the discriminant of an integral basis can be considered as an invariant of the ring $\mathcal{O}_K$. Denote it by $\mathrm{disc}(\mathcal{O}_K)$.

Theorem 1.18. *Let K be a number field with its ring of integers $\mathcal{O}_K$. Let $\{\vartheta_1, \ldots \vartheta_n\}$ and $\{\mu_1, \ldots, \mu_n\}$ be two integral bases of $\mathcal{O}_K$. Then*

$$disc(\vartheta_1, \ldots \vartheta_n) = disc(\mu_1, \ldots, \mu_n).$$

Proof. Writing the ϑ's in terms of the μ's, we have

$$\begin{pmatrix} \vartheta_1 \\ \vdots \\ \vartheta_n \end{pmatrix} = M \cdot \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix}$$

where $M \in \mathbb{Z}^{n\times n}$. Applying each σ_j (embeddings of K in $\mathbb{C}$) to each of the n equations yields the matrix equation

$$\begin{pmatrix} \sigma_1(\vartheta_1) & \cdots & \sigma_n(\vartheta_1) \\ \vdots & \ddots & \vdots \\ \sigma_1(\vartheta_n) & \cdots & \sigma_n(\vartheta_n) \end{pmatrix} = M \cdot \begin{pmatrix} \sigma_1(\mu_1) & \cdots & \sigma_n(\mu_1) \\ \vdots & \ddots & \vdots \\ \sigma_1(\mu_n) & \cdots & \sigma_n(\mu_n) \end{pmatrix}.$$

Taking second power of the determinants, we obtain

$$\operatorname{disc}(\vartheta_1, \dots \vartheta_n) = \det(M)^2 \operatorname{disc}(\mu_1, \dots, \mu_n).$$

Clearly $\det(M)$ and the discriminants are in $\mathbb{Z}$; this shows that $\operatorname{disc}(\mu_1, \dots, \mu_n)$ is a divisor of $\operatorname{disc}(\vartheta_1, \dots \vartheta_n)$, and both have the same sign. On the other hand, a similar argument shows that $\operatorname{disc}(\vartheta_1, \dots \vartheta_n)$ is a divisor of $\operatorname{disc}(\mu_1, \dots, \mu_n)$. To conclude, the discriminants are equal. $\square$

The following two lemmas are used later to prove the theorems.

Lemma 1.19. *Let R be an integral domain and S an integral domain containing R. Let I be an ideal of S, elements $\gamma \neq 0$ be the root of a non-zero polynomial $f(x) \in R[x]$ and suppose that $\gamma \in I$. Then $I \cap R \neq 0$.*

Proof. Because S is an integral domain, we may factor out any powers of x dividing $f(x)$, and can therefore assume that $f(0) \neq 0$. But $\gamma \mid (f(\gamma) - f(0))$, so $-f(0) = f(\gamma) - f(0) \in I \cap R$, as desired. $\square$

Lemma 1.20. *Let I be a non-zero ideal in a ring of integers $\mathcal{O}_K$ of the number field K. Then the ring $\mathcal{O}_K/I$ is finite.*

Proof. $\mathcal{O}_K$ is a free Abelian group with the rank $n = [K : \mathbb{Q}]$. Let $\{\vartheta_1, \dots, \vartheta_n\}$ be a basis of $\mathcal{O}_K$. If we find $m \in \mathbb{Z}$ such that $m \in I$, then $(m) \subseteq I$ therefore $\mathcal{O}_K/I \subseteq \mathcal{O}_K/(m)$. Also we have $\mathcal{O}_K/(m) = \{\sum_{i=1}^n a_i\vartheta_i; a_i \in \mathbb{Z}_m\}$. In the previous lemma we proved that every non-zero ideal I contains a natural number. $\square$

1.2 Dedekind Domains

Definition 1.21. *A **Dedekind domain** is an integral domain R satisfying the following three conditions:*

(i) R is a Noetherian ring;

(ii) Every nonzero prime ideal of R is maximal;

(iii) R is integrally closed.

The condition (iii) means that if x is a root of a monic polynomial with coefficients in R and if x is in the fraction field of R, then in fact $x \in R$.

The condition (i) is equivalent to the each of the following:

1. Every ideal is finitely generated;

2. Every non-empty set of the ideals has a maximal element.

The main reason of dealing with Dedekind domains is the following theorem.

Theorem 1.22. *The ring of integers $\mathcal{O}_K$ in an algebraic number field K is Dedekind.*

Proof. We need to prove that all three conditions of a Dedekind domain are met. First two properties follow from the Lemma 1.20.

(i) For $\mathcal{O}_K$ to be Noetherian means that any ascending chain of ideals stabilizes. But if I is a non-zero ideal, $\mathcal{O}_K/I$ is finite, and the ideals containing I in $\mathcal{O}_K$ are in bijection with the ideals of $\mathcal{O}_K/I$, hence there can only be a finite amount.

(ii) Let P be a non-zero prime ideal of $\mathcal{O}_K$. We need to show that $\mathcal{O}_K/P$ is a field. Following the Lemma 1.20, $\mathcal{O}_K/P$ is a finite ring and as P is prime ideal, the ring is the integral domain. As it is also a finite integral domain, we get a field (for $a \in (\mathcal{O}_K/P)^*$ we define $\varphi_a : (\mathcal{O}_K/P)^* \to (\mathcal{O}_K/P)^*$, $\varphi_a(b) = ab$. This map is injective and thanks to finiteness it is also surjective, therefore there exist $c \in \mathcal{O}_K/P$, such that $ac = 1$).

(iii) Let T be a fraction field of $\mathcal{O}_K$, $\alpha \in T$ and there exists monic irreducible polynomial $f = \sum_{i=1}^{k} \vartheta_i x^i \in \mathcal{O}_K[x], f(\alpha) = 0$. We need to show that $\alpha \in \mathcal{O}_K$. To achieve this we only need to find monic polynomial over $\mathbb{Z}$, such that α is a root. It holds that $\mathcal{O}_K[\vartheta_1, \dots, \vartheta_k, \alpha]$ is a finitely generated ring over $\mathcal{O}_K$ therefore it is a finitely generated ring over $\mathbb{Z}$. Now we can use Lemma 1.5 to find the monic polynomial over $\mathbb{Z}$ with a root α.

□

The most important property of Dedekind domains is the unique factorization of the ideals. First, we cover required theory and then prove this important property.

Note. Recall that if $I_1, \dots, I_n$ are ideals, then their product $I_1 \cdot \ldots \cdot I_n$ is the set of all finite sums $\sum_i a_{1i} a_{2i} \dots a_{ni}$, where $a_{ki} \in I_k$, $k = 1, \dots, n$. It also applies that $I_1 \cdot \ldots \cdot I_n$ is an ideal contained in each I_j. Moreover, if a prime ideal P contains a product $I_1 \cdot \ldots \cdot I_n$ of ideals, then P contains I_j for some j.

Lemma 1.23. *Every non-zero ideal of a Dedekind domain R contains a product of prime ideals.*

Proof. Assume the contrary. If L is the collection of all non-zero ideals that do not contain a product of prime ideals, then, as R is Noetherian, L has a maximal element J, and J cannot be a prime because it belongs to L. Thus there are elements $a, b \in R$ such that $a \notin J, b \notin J$, and $ab \in J$. By maximality of J, the ideals $J + Ra$ and $J + Rb$ each contain a product of prime ideals, hence so does $(J + Ra)(J + Rb) \subseteq J + Rab = J$. This is a contradiction. □

Definition 1.24. *Let R be an integral domain with fraction field T, and let I be an R-submodule of T. We say that I is a* ***fractional ideal*** *of R if $rI \subseteq R$ for some non-zero $r \in R$. We call r a* ***denominator*** *of I. An ordinary ideal of R is a fractional ideal (take $r = 1$), and is sometimes referred to as an* ***integral ideal****.*

We show some properties of the fractional ideal in the following lemma. Proof of this lemma can be found in [6].

Lemma 1.25. *Let R be an integral domain with fraction field T.*

(i) If I is a finitely generated R-submodule of T, then I is a fractional ideal.

(ii) If R is a Noetherian and I is a fractional ideal of R, then I is a finitely generated R-submodule of T.

(iii) If I and J are fractional ideals with denominators r and s, respectively, then $I \cap J$, $I + J$ and IJ are fractional ideals with respective denominators r (or s), rs and rs.

Lemma 1.26. *Let I be a non-zero prime ideal of the Dedekind domain R, T a fraction field of R and let J be the set of all elements $x \in T$ such that $xI \subseteq R$. Then $R \subset J$.*

Proof. Since $RI \subseteq R$, it follows that R is a subset of J. Pick a non-zero element $a \in I$, so that I contains the principal ideal Ra. Let n be the smallest positive integer such that Ra contains a product $P_1 \cdot \ldots \cdot P_n$ of n non-zero prime ideals. Since R is Noetherian, there is such an n by Lemma 1.23, and by the note before, I contains one of the P_i, say P_1. But in a Dedekind domain, every non-zero prime ideal is maximal, so $I = P_1$. Assuming $n \geq 2$, set $I_1 = P_2 \cdot \ldots \cdot P_n$, so that $Ra \not\supseteq I_1$ by minimality of n. Choose $b \in I_1$ with $b \notin Ra$. Now $II_1 = P_1 \cdots P_n \subseteq Ra$, in particular, $Ib \subseteq Ra$, hence $Iba^{-1} \subseteq R$. Element a has an inverse in K but not necessarily in R. Thus $ba^{-1} \in J$ and $ba^{-1} \notin R$. Because $b \in Ra$, it contradicts the choice of b.

The case $n = 1$ must be handled separately. In this case, $P_1 = I \supseteq Ra \supseteq P_1$, so $I = Ra$. Thus Ra is a proper ideal, and we can choose $b \in R$ with $b \notin Ra$. Then $ba^{-1} \notin R$, but $ba^{-1}I = ba^{-1}Ra = bR \subseteq R$, so $ba^{-1} \in J$. □

Proposition 1.27. *Let I be a non-zero prime ideal of the Dedekind domain R, and let $J = \{x \in T : xI \subseteq R\}$. Then J is a fractional ideal and $IJ = R$.*

Proof. If r is a non-zero element of I and $x \in J$, then $rx \in R$, so $rJ \subseteq R$ and J is a fractional ideal. Now $IJ \subseteq R$ by definition of J, so IJ is an integral ideal. Using 1.26, we have $I = IR \subseteq IJ \subseteq R$, and maximality of I implies that either $IJ = I$ or $IJ = R$. In the latter case, the lemma is proved, so assume $IJ = I$.

If $x \in J$, then $xI \subseteq IJ = I$, and by induction, $x^n I \subseteq I$ for all $n \geq 1$. Let r be any non-zero element of I. Then $rx^n \in x^n I \subseteq I \subseteq R$, so $R[x]$ is a fractional ideal. Since R is Noetherian, part (ii) of the 1.25 implies that $R[x]$ is a finitely generated R-submodule of T. From the equivalence of integral element and finitely generated submodule we have that x is integral over R. But R, a Dedekind domain, is integrally closed, so $x \in R$. Therefore $J \subseteq R$, contradicting 1.26. □

Finally, we can now prove the unique factorization of ideals in the Dedekind domain.

Theorem 1.28. *If I is a non-zero fractional ideal of the Dedekind domain R, then I can be factored uniquely as $P_1^{e_1} P_2^{e_2} \cdot \ldots \cdot P_r^{e_r}$, where e_i are integers. Consequently, the non-zero fractional ideals form a group under multiplication.*

Proof. First consider the existence of such a factorization. Without loss of generality, we can restrict to integral ideals. (Note that if $r \neq 0$ and $rI \subseteq R$, then $I = (rR)^{-1}(rI)$.) By convention, we regard R as the product of the empty collection of prime ideals, so let L be the set of all non-zero proper ideals of R that cannot be factored in the given form, with all e_i positive integers. This trick yields the useful result that the factorization of integral ideals only involves positive exponents. Since R is Noerherian, L, if non-empty, has a maximal element I_0, which is contained in a maximal ideal I. By Proposition 1.27, I has an inverse fractional ideal J. Thus by Lemma 1.26 and Proposition 1.27,

$$I_0 = I_0R \subseteq I_0J \subseteq IJ = R. \tag{1.2}$$

Therefore I_0J is an integral ideal, and we claim that $I_0 \subset I_0J$. If $I_0 = I_0J$, then the last paragraph of the proof of 1.27 can be reproduced with I replaced by I_0 to reach a contradiction. By maximality of I_0, I_0J is a product of prime ideals, say $I_0J = P_1 \cdot \ldots \cdot P_r$ (with repetition allowed). Multiply both sides by the prime ideal I to conclude that I_0 is a product of prime ideals, contradicting $I_0 \in L$. Thus L must be empty, and the existence of the desired factorization is established.

To prove uniqueness, suppose that we have two prime factorizations

$$P_1^{e_1} \cdot \ldots \cdot P_r^{e_r} = Q_1^{t_1} \cdot \ldots \cdot Q_s^{t_s} \tag{1.3}$$

where again we may assume without loss of generality that all exponents are positive. If P^{-e} appears, multiply both sides by P^e. Now P_1 contains the product of the $P_i^{e_i}$, so by the note before 1.23, P_1 contains Q_j for some j. By maximality of Q_j, $P_1 = Q_j$, and we may renumber so that $P_1 = Q_1$. Multiply by the inverse of P_1, and continue inductively. □

Proposition 1.29. *Let I be a non-zero fractional ideal of the Dedekind domain R. Then there is a non-zero integral ideal J such that IJ is a principal ideal of R.*

Proof. By Theorem 1.28, there is a non-zero fractional ideal I' such that $II' = R$. By definition of fractional ideal, there is a non-zero element $r \in R$ such that rI' is an integral ideal. If $J = rI'$, then $IJ = Rr$, a principal ideal of R. □

Definition 1.30. *Let I_1 and I_2 be the ideals of the Dedekind domain R. We say that I_1 divides I_2 if $I_2 = I_1J$ for some ideal $J \subseteq R$. We denote it by $I_1|I_2$.*

Lemma 1.31. *Let I_1, I_2 and J be the ideals in the Dedekind domain R. If $JI_1 = JI_2$, then $I_1 = I_2$.*

Proof. Following the Proposition 1.29 there exist the ideal D such that $JD = (a)$, $a \in R$. As $DJI_1 = DJI_2$, we have $aI_1 = aI_2$. Moreover, there exist the bijection between I_1 and aI_1 and between aI_2 and I_2, consequently $I_1 = I_2$. □

Lemma 1.32. *If I and J are ideals in the Dedekind domain R. Then $I|J$ if and only if $J \subseteq I$.*

Proof. If $I|J$, then $J \subseteq I$ is trivial. Let $J \subseteq I$. Following the Proposition 1.29 for the ideal I there exists ideal D such that $ID = (a)$, $a \in R$. As $(a) = ID \supseteq JD$, then the set $C = a^{-1}DJ \subseteq R$. Moreover C is the ideal and $IC = J$. □

The prime factorization of ideals in a Dedekind domain and the principle of dividing ideals give us the way of how to compute the greatest common divisor (gcd) and the least common multiple (lcm) for the ideals. The greatest common divisor is the smallest ideal containing both I and J, that is, $I + J$. The least common multiple is the largest ideal contained in both I and J, which is $I \cap J$.

1.3 Factorization of Ideals to Prime Ideals

We already know that there exists a unique factorization of ideals in a Dedekind domain. Now we have two main goals. First, how to find the prime ideals in a ring of integers. Second, how to find the factorization of ideals to the prime ideals.

In this section, let us denote by K, L the number field, R, S their rings of integers and P, Q their prime ideals. By a prime ideal we mean a nonzero prime ideal. Let us have the following situation. We have a prime ideal P of R and the number fields K, L such that $K \subset L$. We want to find the prime factorization of PS in S.

This next theorem gives the basic properties of the prime ideals. The proofs to the theorems can be found in [5], in Chapter 3.

Theorem 1.33. *Let P be a prime ideal of R and Q a prime ideal of S. Then the following conditions are equivalent:*

(i) $Q|PS$

(ii) $PS \subseteq Q$

(iii) $P \subseteq Q$

(iv) $P = Q \cap R$

(v) $P = Q \cap K$.

When $Q|PS$, we say that Q lies over P, or P lies under Q. The prime ideals lying over a given P are the ones which occur in the prime decomposition of PS.

Definition 1.34. ***Ramification index*** *e is the exact power of a prime ideal Q dividing PS ($Q^e|PS$), where P is a prime ideal of R. We denote the ramification index by* $\mathrm{e}(Q|P)$.
We say that P is ramified in S if $\mathrm{e}(Q|P) > 1$ *for some Q.*

We also know that the factor rings R/P and S/Q are fields since P and Q are maximal ideals. Moreover, fields are finite by Lemma 1.20. Further, field $R/P = R/(P \cap Q) \cong (R + Q)/Q$ can be viewed as a subfield of S/Q, hence S/Q is an extension of a finite degree over R/P.

Definition 1.35. *The* ***inertial degree*** *f is a dimension of S/Q over R/P. We denote it by* $\mathrm{f}(Q|P)$.

If $P \subseteq Q \subseteq U$ are prime ideals in three rings of integers $R \subseteq S \subseteq T$, then

$$\mathrm{e}(U|P) = \mathrm{e}(U|Q) \cdot \mathrm{e}(Q|P)$$
$$\mathrm{f}(U|P) = \mathrm{f}(U|Q) \cdot \mathrm{f}(Q|P).$$

Definition 1.36. *The norm of the ideal I in R is $\mathcal{N}(I) = |R/I|$. Sometimes it is denoted by $||I||$ or $\mathcal{N}_R(I)$.*

The norm is important as it can be used to compute the divisors of I. Let us first show some properties of the norm.

Theorem 1.37. *Let K and L be the number fields, R and S their rings of integers, $K \subseteq L$ and $n = [L : K]$.*

(i) For ideals I and J in R, $\mathcal{N}(IJ) = \mathcal{N}(I) \cdot \mathcal{N}(J)$.

(ii) Let I be an ideal in R. For the S-ideal IS, $\mathcal{N}_S(IS) = (\mathcal{N}_R(I))^n$.

Proof. (i) We prove this first for the case in which I and J are relatively prime, and then show that $\mathcal{N}(P^m) = (\mathcal{N}(P))^m$ for all prime ideals P. This then implies

$$\mathcal{N}(P_1^{m_1} \cdot \ldots \cdot P_r^{m_r}) = \mathcal{N}(P_1^{m_1}) \cdot \ldots \cdot \mathcal{N}(P_r^{m_r}).$$

Factoring I and J into primes and applying the formula above, we obtain the proposition (i).

First we assume that I and J are relatively prime. Then $I + J = R$ and $I \cap J = IJ$. By the Chinese Remainder Theorem we have an isomorphism

$$R/IJ \cong R/I \times R/J,$$

hence,

$$\mathcal{N}(IJ) = \mathcal{N}(I) \cdot \mathcal{N}(J).$$

Next consider $\mathcal{N}(P^m)$, P a prime ideal. We show that $\mathcal{N}(P^m) = (\mathcal{N}(P))^m$. We have a chain of ideals $R \supset P \supset P^2 \supset \ldots \supset P^m$, hence it will be sufficient to show that for each k, $\mathcal{N}(P) = |P^k/P^{k+1}|$ because $\mathcal{N}(P^m) = |R/P^m| = |R/P| \cdot |P/P^2| \cdot \ldots \cdot |P^{m-1}/P^m| = (\mathcal{N}(P))^m$. The ideals P^k are just considered as additive groups. Fixing any $\vartheta \in P^k \setminus P^{k+1}$, we have a group-isomorphism

$$R/P \cong \vartheta R/\vartheta P.$$

Next, the inclusion $\vartheta R = (\vartheta) \subset P^k$ induces the homomorphism

$$(\vartheta) \to P^k/P^{k+1}$$

whose kernel is $(\vartheta) \cap P^{k+1}$ and whose image is $((\vartheta) + P^{k+1})/P^{k+1}$. The element ϑ was chosen such that P^k is the exact power of P dividing (ϑ). Then

$$\begin{aligned}(\vartheta) \cap P^{k+1} &= \operatorname{lcm}((\vartheta), P^{k+1}) = \vartheta P,\\ (\vartheta) + P^{k+1} &= \gcd((\vartheta), P^{k+1}) = P^k.\end{aligned}$$

In conclusion, we get group-isomorphism

$$R/P \cong \vartheta R/\vartheta P \cong P^k/P^{k+1},$$

therefore $\mathcal{N}(P) = |P^k/P^{k+1}|$.

(ii) In view of (i), it is sufficient to prove this for any prime ideal $P \subseteq R$. We can see that S/PS is a vector space over the field R/P. We claim that its dimension is n. First we show that the dimension is at most n. It is sufficient to prove that any $n+1$ elements are linearly dependent over R/P. We denote these elements by $\mu_1, \ldots, \mu_{n+1} \in S$. They are linearly dependent over K, and it follows that they are linearly dependent over R. Thus we have $\vartheta_1 \cdot \mu_1 + \ldots + \vartheta_{n+1} \cdot \mu_{n+1} = 0$ for some $\vartheta_i \in R$, not all 0. We need to show that at least one ϑ_i does not lie in P, therefore we get that they are linearly dependent over R/P. We prove this by contrary. Let all $(n+1)$-tuples $\{\vartheta_1, \ldots, \vartheta_{n+1}\} \subseteq P$. Arbitrarily we choose one of them and consider the ideal $(\vartheta_1, \ldots, \vartheta_{n+1}) \subseteq P$. Following the Proposition 1.29 there exists ideal I such that $(\vartheta_1, \ldots, \vartheta_{n+1}) \cdot I$ is principal ideal. We denote it by (ϑ). Then $(\vartheta_1, \ldots, \vartheta_{n+1}) \cdot I = \vartheta R \nsubseteq \vartheta P$ as $P \subsetneq R$. We choose $\xi \in I$ such that $\xi(\vartheta_1, \ldots, \vartheta_{n+1}) \nsubseteq \vartheta P$. So $\frac{\xi}{\vartheta}(\vartheta_1, \ldots, \vartheta_{n+1}) \nsubseteq P$, but also $\frac{\xi}{\vartheta}(\vartheta_1, \ldots, \vartheta_{n+1}) \subseteq R$ which is the contradiction. The dimension is therefore at most n.

To establish equality, let $P \cap \mathbb{Z} = p\mathbb{Z}$ and consider all prime ideals P_i of R lying over prime p. We know S/P_iS is a vector space over R/P_i of dimension $n_i \leq n$; we will show that equality holds for all i, hence in particular when $P_i = P$. Set $e_i = \mathrm{e}(P_i|(p))$ and $f_i = \mathrm{f}(P_i|(p))$. Let m be the degree of K over $\mathbb{Q}$. Following the special case of Theorem 21 in [5] we have $\mathcal{N}(pR) = p^m$ and

$$p^m = \mathcal{N}(pR) = \mathcal{N}(P_1^{e_1} \cdot \ldots \cdot P_r^{e_r}) = \prod_{i=1}^{r} \mathcal{N}(P_i)^{e_i} = \prod_{i=1}^{r} p^{f_i e_i}.$$

We have $pR = \prod_{i=1}^{r}(P_i)^{e_i}$, hence $pS = \prod_{i=1}^{r}(P_iS)^{e_i}$ and from the above we obtain

$$\mathcal{N}(pS) = \prod_{i=1}^{r} \mathcal{N}(P_iS)^{e_i} = \prod_{i=1}^{r} \mathcal{N}(P_i)^{n_i e_i} = \prod_{i=1}^{r} (p^{f_i})^{n_i e_i}.$$

On the other hand we know that $\mathcal{N}(pS) = p^{mn}$, so $mn = \sum_{i=1}^{r} f_i n_i e_i$. Since all $n_i \leq n$ and $\sum_{i=1}^{r} f_i e_i = m$, it follows that $n_i = n$ for all i. $\square$

The previous theorem gives us some restrictions on the possible divisors of the ideal I if we know its norm. Point (ii) shows that whenever $\mathcal{N}(I)$ is a prime, I must be a prime ideal.

The following theorem defines the different way of computing the norm of the principal ideal.

Theorem 1.38. *Let K be a number field and R its ring of integers. Let $\vartheta \in R$, $\vartheta \neq 0$. For the principal ideal (ϑ),*

$$\mathcal{N}((\vartheta)) = |N_{\mathbb{Q}}^{K}(\vartheta)|.$$

Proof. Extend K to a normal extension M of $\mathbb{Q}$, and let T be the ring of integers of M. For each embedding σ of K in $\mathbb{C}$, we have:

$$\mathcal{N}(\sigma(\vartheta)T) = \mathcal{N}(\vartheta T).$$

If we extend σ to an automorphism of M as then $\sigma(T) = T$. Set $a = N_{\mathbb{Q}}^{K}(\vartheta) = \prod_{i=1}^{n} \sigma_i(\vartheta) \in \mathbb{Z}$. Then by 1.37 (i) we have:

$$\mathcal{N}(aT) = \prod_{i=1}^{n} \mathcal{N}(\sigma_i(\vartheta)T) = \mathcal{N}(\vartheta T)^n.$$

Clearly $\mathcal{N}(aT) = |a|^{mn}$ where $m = [M : K]$, and 1.37 (ii) shows that $\mathcal{N}(\vartheta T) = \mathcal{N}(\vartheta R)^m$. Putting this together we obtain $|a| = \mathcal{N}(\vartheta R)$. $\square$

The next theorem shows the fundamental property in the number fields.

Theorem 1.39. *Let R and S be the rings of integers of the number fields K and L respectively, and $n = [L : K]$. Let P be a nonzero prime ideal of R and*

$$PS = Q_1^{e_1} \cdot \ldots \cdot Q_k^{e_k},$$

with Q_i the prime ideals of S and $f_i = \mathrm{f}(Q_i|P)$. Then

$$n = \sum_{i=1}^{k} e_i f_i.$$

Proof. Following the Lemma 1.37 we have $\mathcal{N}(PS) = \mathcal{N}(P)^n$ and

$$\mathcal{N}(PS) = \mathcal{N}(\prod_{i=1}^{k} Q_i^{e_i}) = \prod_{i=1}^{k} \mathcal{N}(Q_i)^{e_1} = \prod_{i=1}^{k} \mathcal{N}(P)^{f_i e_i}.$$

Thus $n = \sum_{i=1}^{k} e_i f_i$. □

We now show two more theorems with other useful properties of divisors of the ideals. Proofs of these can be found in [5], Chapter 3.

Theorem 1.40. *Let $K = \mathbb{Q}$, $R = \mathbb{Z}$ and L be a number field with ring of integers S. If $p \in \mathbb{Z}$ is a prime number, then principal ideal $p\mathbb{Z} \subseteq \mathbb{Z}$ is ramified in S if and only if $p | disc(S)$.*

Theorem 1.40 states that there exists only a finite amount of ramified prime ideals $P = p\mathbb{Z}$. So PS is the product of prime ideals in the first power (except of some finite number of other cases). Also, Theorem 1.39 gives the number of these possible prime ideals.

The next theorem defines more possibilities of how to determine the divisors of PS for almost all prime numbers.

Theorem 1.41. *Let $K \subseteq L$ be the finite fields with their rings of integers R and S, $n = [L : K]$ and let $L = K[\vartheta]$ for some $\vartheta \in S$ with monic irreducible polynomial h over K. Let P be a prime ideal of R lying over prime number p, where p does not divide $|S/R(\vartheta)|$. Moreover, let*

$$h \equiv h_1^{e_1} \cdot \ldots \cdot h_k^{e_k} \ mod\ P[x].$$

Then

$$PS = Q_1^{e_1} \cdot \ldots \cdot Q_k^{e_k},$$

where $Q_i = PS + h_i(\vartheta)S = (P, h_i(\vartheta))$. Also, $\mathrm{f}(Q_i|P) = deg\ h_i$.

Prime numbers that do not divide the order of the factor group $S/R[\vartheta]$ are called *non-special prime numbers*. We just showed how the divisors of ideal PS look like if a prime ideal P is over a non-special prime number p. More complicated case is with the *special prime numbers*, those dividing $|S/R[\vartheta]|$. A field extension $R[\vartheta]$ has to be created in a way that there lies an ideal which is the multiplication of all prime ideals lying over a special prime number p. We do not cover this in our work as it requires to define a lot of prerequisite theorems and definitions. We recommend Chapter 6 in [7] for more details.

Lemma 1.42. *Let $K = \mathbb{Q}[\theta]$ and (a, b) be coprime integers. Then any prime ideal P which divides $(a - b\theta)$, either divides the index $n = [\mathcal{O}_K : \mathbb{Z}[\theta]]$ or is of degree one.*

Proof. Let P be a prime ideal above a prime number p. Then $p \nmid b$ otherwise $a \in P \cap \mathbb{Z}$ hence $p \mid a$, contradicting a and b being coprime. Now assume that $p \nmid n$, and let b^{-1} be an inverse of b modulo p and u be an inverse of n modulo p. We have $\theta \equiv -ab^{-1} \pmod{P}$. If $x \in \mathcal{O}_K$, $nx \in \mathbb{Z}[\theta]$ so there exists a polynomial $f \in \mathbb{Z}[X]$ such that $x \equiv uf(-ab^{-1}) \pmod{P}$ so any element of $\mathcal{O}_K$ is congruent to a rational integer modulo P, hence to an element of the set $\{0, 1, \ldots, p-1\}$. $\square$

1.4 Units in Number Fields

A unit u in K is an algebraic integer, such that $1/u$ is also an algebraic integer. Equivalently, following the Lemma 1.11, unit u in K is an algebraic integer of norm ± 1.

Definition 1.43. *The **set of units** in K forms a multiplicative group which we denote by $\mathcal{O}_K^*$. The torsion subgroup of $\mathcal{O}_K^*$, i.e., **the group of roots of unity** in K, is denoted by $\mu(K)$.*

We now first define the signature of a number field and then describe the properties of a group $\mathcal{O}_K^*$ using the signature.

Definition 1.44. *The **signature** of a number field K is the pair (r_1, r_2), where r_1 is the number of embeddings of K whose images lie in R, and $2r_2$ is the number of non-real complex embeddings, so that $r_1 + 2r_2 = n$ and $n = [K : \mathbb{Q}]$. If f is an irreducible polynomial defining the number field K by one of its roots, the signature of K is also called the signature of f. Here r_1 is the number of real roots of f in $\mathbb{C}$. Analogically for $2r_2$.*

One way to determine the signature of the number field K is to use the defining polynomial. If $K = \mathbb{Q}[\theta]$ and f is the minimal polynomial of θ, we can compute the roots of f in $\mathbb{C}$ and count the number of real roots. Another way is to use Sturm's theorem which uses the sequence of leading coefficients in the polynomial remainder sequence obtained by applying Euclid's algorithm, see Theorem 4.1.10 in [7].

Theorem 1.45. *Let (r_1, r_2) be the signature of K. Then the group $\mathcal{O}_K^*$ is a finitely generated Abelian group of rank $r_1 + r_2 - 1$. In other words, we have a group isomorphism*

$$\mathcal{O}_K^* \simeq \mu(K) \times \mathbb{Z}^{r_1 + r_2 - 1}, \tag{1.4}$$

and $\mu(K)$ is a finite cyclic group.

If we set $r = r_1 + r_2 - 1$, then there exist units $u_1, \ldots, u_r$ such that every element $u \in \mathcal{O}_K^*$ can be written in a unique way as

$$u = \vartheta u_1^{n_1} u_2^{n_2} \cdot \ldots \cdot u_r^{n_r}, \tag{1.5}$$

where $n_i \in \mathbb{Z}$ and ϑ is a root of unity in K. Set $(u_1, u_2, \ldots, u_r)$ is called the *system of fundamental units* of K.

We now linearize the multiplicative group of units by taking the logarithms. Let $\sigma_1, \ldots, \sigma_{r_1}, \sigma_{r+1}, \ldots, \sigma_{r_1+r_2}$ be the first $r_1 + r_2$ embeddings of K in $\mathbb{C}$ where the σ_i for $i \leq r_1$ are the real embeddings, and the other embeddings are the σ_i and $\bar{\sigma}_i = \sigma_{r_2+1}$ for $i > r_1$.

Definition 1.46. *The logarithmic embedding of $\mathcal{O}_K^*$ in $\mathbb{R}^{r_1+r_2}$ is the map L which sends x to*

$$L(x) = (\ln|\sigma_1(x)|, \ldots, \ln|\sigma_{r_1}(x)|, 2\ln|\sigma_{r_1+1}(x)|, \ldots, 2\ln|\sigma_{r_1+r_2}(x)|).$$

L is an Abelian group homomorphism.

1.5 Miscellaneous

In the previous section we used the logarithm to linearize the group of units. We use the same principle in our examples in this work for the group $(\mathbb{Z}/p\mathbb{Z})^*$. We know that the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$ is isomorphic to the (additive) group $\mathbb{Z}/(p-1)\mathbb{Z}$, therefore if we have the equation

$$x \cdot y \equiv z \pmod{p}$$

and we take the logarithm of both sides, we obtain

$$\log(x \cdot y) = \log(x) + \log(y) \equiv \log(z) \pmod{p-1}.$$

Another very important definition is the smoothness of the elements. We need to represent the elements where their divisors have some specified size.

Definition 1.47. *Let x and B be natural numbers. We say that x is B-**smooth** if all prime divisors of x are smaller than B.*

We also define the L-notation of the complexity that we use for the number field sieve algorithms.

$$L_q[\alpha, c] = \exp((c + o(1))(\log q)^\alpha (\log\log q)^{1-\alpha}), \tag{1.6}$$

where $q = p^n$, c is a constant, $0 \leq \alpha \leq 1$ and $q \to \infty$.

At the end, let us define the discrete logarithm, where the discrete logarithm problem is the mathematical problem this work is devoted to.

Definition 1.48. ***Discrete logarithm*** *is an integer x solving the equation $t^x \equiv u$ (mod p), where p is a prime number and x is denoted by $x = \log_t(u)$.*

Chapter 2

Index Calculus

This chapter discusses the Index Calculus algorithm that computes the discrete logarithm, with an example using integers to illustrate the problem. We analyze the number field sieve algorithm, which is based on the Index Calculus algorithm, in the next chapter.

Index Calculus algorithm was designed to compute the discrete logarithm. It was first described by Adleman in [3]. It is based on the principle of computing discrete logarithms of some small elements first and then computing the desired discrete logarithm by using these precomputed logarithms. This method uses two well known properties of the logarithm:

$$\begin{aligned} \log_g(a \cdot b) &= \log_g(a) + \log_g(b), \text{ and} \\ \log_g(a^e) &= e \cdot \log_g(a). \end{aligned}$$

So if we know that $\log_g(u) = v$ and factorization of u is $u = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$, then v can be written as

$$v = \log_g(u) = e_1 \log_g(p_1) + e_2 \log_g(p_2) + \ldots + e_k \log_g(p_k).$$

The Index Calculus algorithm is usually computed in a group of the form $\mathbb{F}_q^*$. The algorithm requires to find the efficient factor base, therefore the groups containing integers or polynomials are mainly used. In contrast, it is impossible to find the efficient factor base in the group of points on elliptic curves.

In next sections, we cover the usage of the algorithm for integers in a basic example, then we describe the algorithm for polynomials.

2.1 Example of Index Calculus Algorithm for Integers

In this section, we show the Index Calculus algorithm on an example. We collect the relations between the discrete logarithms of small elements from some factor base, solve them using linear algebra and then compute the searched discrete logarithm with respect to these precomputed discrete logarithms.

Before we continue, we clarify the usage of the logarithm in this section. When we have the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$ then the logarithm is a group homomorphism of this group to the additive group $\mathbb{Z}/(p-1)\mathbb{Z}$. Therefore when computing the discrete logarithm x such that $t^x \equiv u \pmod{p}$ in some group $\mathbb{F}_p^* = \mathbb{Z}_p^*$, by applying the logarithm we get the equation $x \equiv \log_t(u) \pmod{p-1}$. For example, if we want to compute

$$3^x \equiv 5 \pmod{7},$$

by applying the logarithm on both sides we get

$$x \equiv \log_3(5) \pmod{6}.$$

We use this isomorphism in our example that is adopted from [8].

Let us have a group $\mathbb{F}_p^*$, where $p = 1009$ is a prime number and the generator of $\mathbb{F}_{1009}^*$ is 11. We want to find x such that

$$11^x \equiv 946 \pmod{1009}.$$

We choose $S = \{2, 3, 5, 7, 11, 13\}$ to be the factor base. As we want to compute the discrete logarithms of these elements, we have 6 unknown variables. In order to determine the values of these unknowns we need at least 6 linearly independent relations. We look for the numbers in a special form defined below, such that these numbers factorize to the elements from our factor base. One relation is clear

$$\log_{11} 11 \equiv 1 \pmod{1008}.$$

To find the remaining relations, random powers of 11 mod 1009 are chosen and checked if they are B-smooth, for $B = 14$. We get the following equations

$$\begin{array}{lll}
11^{300} = 112 \pmod{1009} & \text{which factorizes as} & 2^4 \cdot 7, \\
11^{23} = 390 \pmod{1009} & \text{which factorizes as} & 2 \cdot 3 \cdot 5 \cdot 13, \\
11^{900} = 400 \pmod{1009} & \text{which factorizes as} & 2^4 \cdot 5^2, \\
11^{134} = 768 \pmod{1009} & \text{which factorizes as} & 2^8 \cdot 3, \\
11^{797} = 165 \pmod{1009} & \text{which factorizes as} & 3 \cdot 5 \cdot 11, \\
11^{43} = 65 \pmod{1009} & \text{which factorizes as} & 5 \cdot 13.
\end{array}$$

Then we apply the logarithm on both sides of the equations to obtain the following relations:

$$\begin{aligned}
\log_{11} 11 &\equiv 1 && \pmod{1008}, \\
4\log_{11} 2 + \log_{11} 7 &\equiv 300 && \pmod{1008}, \\
\log_{11} 2 + \log_{11} 3 + \log_{11} 5 + \log_{11} 13 &\equiv 23 && \pmod{1008}, \\
4\log_{11} 2 + 2\log_{11} 5 &\equiv 900 && \pmod{1008}, \\
8\log_{11} 2 + \log_{11} 3 &\equiv 134 && \pmod{1008}, \\
\log_{11} 3 + \log_{11} 5 + \log_{11} 11 &\equiv 797 && \pmod{1008}, \\
\log_{11} 5 + \log_{11} 13 &\equiv 43 && \pmod{1008}.
\end{aligned}$$

From the congruences we compose a matrix M

$$M = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{cccccc} \log_{11}(2) & \log_{11}(3) & \log_{11}(5) & \log_{11}(7) & \log_{11}(11) & \log_{11}(13) \\ \end{array}$$

$$M = \left(\begin{array}{cccccc} 0 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 4 & 0 & 2 & 0 & 0 & 0 \\ 8 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right)$$

The right-side vector is

$$C = (1, 300, 23, 900, 134, 797, 43)^T.$$

By solving equation $MX = C$ we get $X = (886, 102, 694, 788, 1, 357)^T$. So the discrete logarithms of the elements from the factor base S are:

$$\begin{array}{lll} \log_{11} 2 = 886, & \log_{11} 3 = 102, & \log_{11} 5 = 694, \\ \log_{11} 7 = 788, & \log_{11} 11 = 1, & \log_{11} 13 = 357. \end{array}$$

Now, we want to compute the searched discrete logarithm using these precomputed logarithms. To achieve this, we need to find a power γ of 11, such that $946 \cdot 11^\gamma$ is 14-smooth. One solution is $\gamma = 491$ because

$$946 \cdot 11^{491} \equiv 624 = 2^4 \cdot 3 \cdot 13 \pmod{1009}.$$

Once we have the solution, we are able to compute the discrete logarithm of x from $11^x \equiv 946 \pmod{1009}$ as we have

$$2^4 \cdot 3 \cdot 13 \pmod{1009} = 946 \cdot 11^{491} \equiv 11^x \cdot 11^{491} = 11^{x+491}$$

and by applying the logarithm on both sides we get

$$4 \log_{11} 2 + \log_{11} 3 + \log_{11} 13 \pmod{1008} \equiv x + 491.$$

Therefore

$$x \equiv 4 \cdot 886 + 102 + 357 - 491 \equiv 488 \pmod{1008}.$$

So the solution of the discrete logarithm is 488.

Using PARI/GP to compute discrete logarithm with Index Calculus algorithm

PARI/GP[1] is a computer algebra system which was designed for fast computations in number theory. It enables direct usage of the most common data types used in the factorization, algebraic number theory, elliptic curves as well as polynomials, matrices, algebraic numbers and many more. It also contains an

[1]`http://pari.math.u-bordeaux.fr`

extensive algebraic number theory module, therefore it is an excellent system to be used when computing the Index Calculus algorithm.

A script implementing the Index Calculus algorithm for $\mathbb{F}_p^*$ written in PARI/GP is a part of this work. It computes the discrete logarithm and uses the built in command `znlog` for verification of the algorithm. The script consists of two main functions:

`ComputeSmallLogarithms` - computes the discrete logarithms of the elements from the factor base S. It is looking for smooth elements from which it produces $3 * |S|$ relations. By composing them together to the matrix and solving this matrix by using the Gaussian elimination, the discrete logarithms are acquired.

`ComputeLogarithm` - computes the desired discrete logarithm with respect to the precomputed small logarithms by finding the smooth element of special form.

At this moment, there are three issues in the algorithm. One is with the matrix composed in `ComputeSmallLogarithms` function. Even if the number of relations is three times the size of the factor base, the rank of matrix is sometimes smaller than the size of factor base. The solution is therefore not unique and the result does not have to be correct. This could be avoided by adding a condition to the code to check if the rank is big enough or if the determinant of the square matrix is non zero. The code includes this condition, but it is commented out as for bigger inputs it could take some time to find the matrix with non zero determinant.

The second issue is the Gaussian elimination. As this solving method for matrices is numerically unstable, caused by the possibility of dividing by very small numbers, it sometimes outputs an incorrect solution even if the determinant is not zero. This is a generally known problem so one could run the script few times and choose the solution with the highest occurrence.

The last issue is the restriction for the algorithm. This script uses the Gaussian elimination which is slow for big matrices, one should use this algorithm for smaller numbers.

The example above shows how the Index Calculus method is used when we are computing in the group $\mathbb{F}_p^*$. In case of $\mathbb{F}_{p^n}^*$, situation is more complicated and a version with polynomials is used.

2.2 Index Calculus Algorithm for $\mathbb{F}_{p^n}$

This section is devoted to the description of the Index Calculus method using polynomials. Let's have a group $\mathbb{F}_{p^n}$ where we want to compute the discrete logarithm. Every element of the group $\mathbb{F}_{p^n}$ could be represented as a polynomial over $\mathbb{F}_p$ of degree at most n. For working with polynomials, we need the following definition to clarify the meaning of the smoothness of the element in the polynomial representation.

Definition 2.1. *Element is **B-smooth**, if its polynomial representation $f \in \mathbb{F}_p[x]$ factorizes to powers of factors that are of degree less than B.*

So if $f = g_1^{e_1} g_2^{e_2} \dots g_k^{e_k}$, g_i irreducible, then $\deg(g_i) < B$ for $i = 1, \dots, k$.

Then we define the factor base, sometimes also called the smoothness base, of smooth elements

$$S = \{g \in \mathbb{F}_p[x] \mid \deg(g) < B, g \text{ irreducible}\}.$$

As mentioned above, the algorithm is composed of two parts. The first part consists of computing the discrete logarithms of all elements from the factor base S. To achieve this, we need to find relations between these discrete logarithms. After obtaining enough relations, we need to solve the system of equations to calculate the discrete logarithms. In the second part, the searched discrete logarithm is computed using the precomputed logarithms. The following paragraph shows how to obtain one relation for the first part.

Choose a random integer $s \in \{1, \ldots, p^n - 1\}$ and compute $g^s \pmod f$ where g is the primitive element and f is a chosen irreducible polynomial of degree n such that $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/(f)$. Verify that the computed polynomial is B-smooth. If not, choose a new value s and compute the polynomial again. In the case of a smooth element, we get

$$g^s \pmod f = g_1^{e_1} g_2^{e_2} \ldots g_k^{e_k} \pmod f,$$

where $g_i \in S$ for $i = 1, \ldots, k$.

After applying the logarithm on both sides, we get

$$s = \sum_{i=1}^{k} e_i \log_g g_i \pmod{p^n - 1},$$

which is exactly the relation of discrete logarithms of some elements from S. This means that, with every successful choice of s, we get one relation. If we get $|S|$ linearly independent relations, we can clearly determine the logarithms of all the elements.

Since the matrix of the relations could be huge but sparse, there exist algorithms for solving such matrices effectively. One of them is Wiedemanns algorithm presented in [9].

In the second part of the algorithm, we use the precomputed logarithm to find the searched logarithm. This consists of a few steps. First, denote by h a polynomial representation of which element we want to compute the discrete logarithm. Then choose a random integer s such that $1 \leq s \leq p^n - 1$ and compute

$$h^* = h \cdot g^s \pmod f.$$

Verify that h^* is B-smooth, i.e., all the factors of h^* are in the set S. If no, choose a new value s. If yes, we get

$$h^* = g_1^{e_1} g_2^{e_2} \ldots g_k^{e_k}$$

and by applying the logarithm we get

$$\log_g(h \cdot g^s) = \sum_{i=1}^{k} e_i \log_g g_i,$$

$$\log_g h + s = \sum_{i=1}^{k} e_i \log_g g_i.$$

So the desired discrete logarithm is

$$\log_g h = \sum_{i=1}^{k} e_i \log_g g_i - s \pmod{p^n - 1}.$$

To summarize, the entire algorithm consists of three steps:

1. Find enough linear relations between the elements of the factor base S.
2. Solve the matrix of relations to obtain the discrete logarithms of elements from S.
3. Compute the searched discrete logarithm using the discrete logarithms of elements from S.

The first two steps do not rely on the discrete logarithm we are looking for, only on the group we are computing in and the elements in factor base. So if we want to compute more than one discrete logarithm in the same group, we could consider the first two steps as precomputation that is done just once.

Chapter 3

Number Field Sieve Algorithm for Discrete Logarithm

The number field sieve algorithm along with the function field sieve algorithm are the asymptotically fastest algorithms solving the discrete logarithm problem. They are both based on the Index Calculus algorithm that has been described in the previous chapter. As it was mentioned in the introduction, it depends on the choice of p and n to decide which algorithm is the most preferable to use. The main difference between the number field sieve and the function field sieve is that with the number field sieve, the objects are numbers in number fields and the smoothness bases contain ideals of a small norm. With the function field sieve, the objects are polynomials in function fields and the smoothness bases contain ideals whose norms are polynomials of a small degree. We cover only the number field sieve algorithm in this work.

This chapter contains the analysis of the number field sieve algorithm solving the discrete logarithm problem. It also covers the examples of this algorithm for $\mathbb{F}_p$ and $\mathbb{F}_{p^n}$.

Background

There was a long process of developing ideas that finally lead to the algorithm today known as the number field sieve. According to Joux and Lercier in [10], the first works can be traced back to 1920s done by Kraitchik in [11] and [12], following by the work of Western and Miller in [13] in 1960s, who used a similar technique as today's Index Calculus method. In late 1970s, Merkle in [14] and Adleman in [3] independently designed the first algorithmic version of these ideas for prime fields $\mathbb{F}_p$. This version is already divided into two main phases, the first phase uses relations to compute indices of small primes modulo p, and the second phase uses a randomized trial and error procedure to determine the logarithm of an arbitrary element from these indices.

The extension to the finite fields $\mathbb{F}_q$, where $q = p^n$ with p fixed and n tending to infinity, was first made by Hellman and Reyneri in [15]. In this variation, the important role of small primes in the Merkle-Adleman's algorithm was played by irreducible polynomials of a low degree. They derived a similar asymptotic running time $L_p(1/2, 1)$ as Merkle and Adleman. Later, Coppersmith came with a much faster algorithm in [16] with impressive expected running time $L_q(1/3)$.

A lot of effort was put to improve the algorithms and to find new ways to obtain a better complexity. For the finite fields with large characteristic, the first step was made by El Gamal in [17] by publishing an algorithm for $\mathbb{F}_{p^2}$ with the complexity $L_q(1/2, 4\sqrt{3})$. El Gamal was also the first who proposed an idea that number fields might be useful to compute the discrete logarithm.

Back to the $\mathbb{F}_p$, in 1986, Coppersmith et al. in [18] introduced Gaussian Integer variation of Index Calculus. Although complexity was still $L_p(1/2, 1)$, it was more efficient in practice than the former algorithms for $\mathbb{F}_p$.

In 1988, Pollard suggested the use of number fields for factorization and later other researchers evolved the general number field sieve, see [19]. At a similar time, Adleman and Demarrais in [20] presented, that in theory, the expected running time $L_q(1/2, c)$ can be achieved for all finite fields $\mathbb{F}_p$. Moreover, they introduced so called character signatures to deal with the obstruction coming from the units and class group of the number fields.

In the same year, Gordon in [1] presented an algorithm for solving the discrete logarithm problem in $\mathbb{F}_p$ considering more general number fields. He also proposed the solution for units group problem with LLL algorithm to find multiplicative relations between the units collected during the sieving phase. Expected running time of his algorithm is $L_p(1/3, 3^{2/3})$.

Joux and Lercier in [10] also mention that shortly after Schirokauer significantly improved this expected running time for $\mathbb{F}_p$ to $L_p(1/3, (64/9)^{1/3})$ in his work [21] by defining l-adic analogues of character signatures (l divides $p-1$), that can be efficiently computed for any number field. Couple of years later, with a slight modification of this algorithm for fixed n, Schirokauer was able to achieve the expected running time $L_q(1/3, (64/9)^{1/3})$ for $\mathbb{F}_q$, see [22].

Further improvements were also achieved in the number field for factorization. For example, Coppersmith in [23] specified a `NFS` algorithm with concurrent usage of several number fields which led to a slightly lowered complexity.

In the discrete logarithm problem, Adleman and Huang in [24] proposed another algorithm with one of the best complexities $L_q(1/3, (32/9)^{1/3})$ focusing on the intermediate case where neither p nor n are fixed.

Several attempts were made to obtain the complexity $L_q(1/3, c)$, one of them by Adleman and Huang in [24] mentioned above.

As a result of different approaches, using the number field sieve and the function field sieve and different values for n and p, Joux and Lercier in [10] came with a summarization of the complexity for these possibilities as follows:

- there exists an `NFS-DLP` variant which runs in expected time $L_q[1/3, (64/9)^{1/3}]$ when $p \gg L_q[2/3, 1]$,
- there exists an `FFS-DLP` variant which runs in expected time $L_q[1/3, (32/9)^{1/3}]$ when $p \ll L_q[1/3, 1]$,
- there exists another `NFS-DLP` variant which runs in expected time $L_q[1/3, (128/9)^{1/3}]$ in the remaining cases.

In this work we present `NFS-DLP` variant, where p is a medium to large prime number and $n > 1$, more precisely, $p = L_{p^n}(2/3, c)$ and c near $2 \cdot (1/3)^{1/3}$, where we refer to L-notation 1.6.

3.1 Number Field Sieve Algorithm

The number field sieve algorithm for discrete logarithm problem is inspired by `NFS` algorithm for factorization motivated by Kraitchik's method. Its description follows. Let's have a number N to factorize. If we can generate pairs (x_i, y_i), such that

$$x_i^2 \equiv y_i^2 \bmod N, \tag{3.1}$$

then $\gcd(x_i - y_i, N)$ is a non-trivial divisor of N with probability $1/2$. After finding enough such pairs, N could be factorized.

Equation 3.1 is usually expanded to look for relations in the form

$$x_i^2 \equiv p_0^{e_0} p_1^{e_1} \dots p_d^{e_d} \bmod N, \tag{3.2}$$

where $\{p_0, p_1, \cdots p_d\}$ are primes chosen beforehand with -1 included. Once a sufficient amount of relations are found, it is possible to compose a pair $(x,\ y)$ and compute $\gcd(x - y, N)$.

A similar approach is used for the discrete logarithm problem as it was illustrated in the Index Calculus method. To recap, when enough relations between elements $p_1, p_2, \dots, p_l$ are found, matrix composed of these relations is solved and by applying the logarithm, we get $\log(p_i) = d_i$. So if one can express some element c as $c = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_l^{e_l}$, then it is easy to compute $\log(c)$ as we have

$$\log(c) = e_1 \log(p_1) + e_2 \log(p_2) + \dots + e_l \log(p_l) = e_1 d_1 + e_2 d_2 + \dots + e_l d_l. \tag{3.3}$$

Now, we describe the number field sieve algorithm for solving the discrete logarithm problem. In the upcoming sections, we will discuss all the steps in more detail.

Given integers t, u, n and a prime number p, $p^n = q$, we want to compute x, such that $t^x \equiv u \pmod q$, i.e $\log_t u \ = \ x \pmod{q-1}$. We assume that t is a generator of a group $\mathbb{F}_q^*$, therefore $u \in \langle t \rangle$.

The first step of the algorithm is to find two monic irreducible polynomials f_1 and f_2 such that there exists $m \in \mathbb{Z}$ with the property

$$f_i(m) = 0 \pmod p. \tag{3.4}$$

Some approaches to choose such polynomials to satisfy the required properties are analyzed in Section 3.2. Denote by $\theta_i \in \mathbb{C}$ the root of the polynomial f_i. Then it holds that f_i is the minimal polynomial of θ_i and we can define number fields as $K_i = \mathbb{Q}[\theta_i]$. This step is important in the number field sieve algorithm. A number field, respectively its ring of integers $\mathcal{O}_{K_i}$ is used to find relations between elements, in our case, between prime ideals. As ring of integers is a Dedekind domain, it ensures the unique factorization of the ideals to prime ideals. The usage of the number fields is graphically described in the diagram 3.1 as it was presented in Joux and Lercier in [10]

$$\begin{tikzcd}
& \mathbb{Q}[X] \arrow[dl, "\mu_1"'] \arrow[dr, "\mu_2"] & \\
\mathbb{Q}[\theta_1] \arrow[dr, "\phi_1"'] & & \mathbb{Q}[\theta_2] \arrow[dl, "\phi_2"] \\
& \mathbb{F}_q &
\end{tikzcd}$$

Figure 3.1: Usage of the number fields

with the homomorphisms

$$\mu_1 : X \to \theta_1,\ \ \mu_2 : X \to \theta_2,\ \ \phi_1 : \theta_1 \to m \text{ and } \phi_2 : \theta_2 \to m.$$

As it was mentioned, we look for the relations between prime fields from the rings of integers $\mathcal{O}_{K_i}$. When enough relations are found and we solve the matrix composed of these relations, we then use the maps ϕ_i to convert the result to $\mathbb{F}_q$.

To find the relations, first we need to choose smoothness bounds B_i, sieve limits T_i and factor bases $S_1 = \{P_1, P_2, \cdots, P_k\}$, $S_2 = \{Q_1, Q_2, \cdots, Q_l\}$ composed of prime elements of the norm less than B_i. Then we are looking for the pairs (a, b) where $a, b \in \mathbb{Z}$, $|a|, |b| \leq T_i$ and $\gcd(a, b) = 1$, such that ideals $(a + b\theta_i)$ are factorized to prime ideals from the factor basis S_i. To achieve this, one chooses a random pair (a, b), computes its norm $N_{K/\mathbb{Q}}(a + b\theta_i)$ and tests whether this norm is B_i-smooth. If it holds, we get

$$(a + b\theta_1) = \prod_{P_j \in S_1} P_j^{e_j} \text{ and} \tag{3.5}$$

$$(a + b\theta_2) = \prod_{Q_j \in S_2} Q_j^{e_j} \tag{3.6}$$

which gives us one relation. When enough relations of this form are found, we then need to transform the ideals back to elements. If the class number of the number field equals one, every ideal I in $\mathcal{O}_{K_i}$ is principal, therefore there exists $\gamma_I \in \mathcal{O}_{K_i}$ with $I = (\gamma_I)$. The ideal decomposition $(a + b\theta_1) = \prod_{P_j \in S_1} P_j^{e_j}$ then leads to

$$a + b\theta_1 = u \cdot \prod_j \gamma_j^{e_j}, \tag{3.7}$$

with $(\gamma_j) = P_j$ and u a unit in $\mathcal{O}_{K_i}$, as it was mentioned in [10]. Similarly for $(a + b\theta_2)$.

A more complicated situation arises when the class number is not equal to one. In the factorization, one adds characters to the relations to ensure feasibility of the square root computation step. In the discrete logarithm, it is more difficult to complete the relations; however, as Joux and Lercier explain in [25], Schirokauer in his work [21] defined so called linear maps that are easily computable, which can efficiently replace characters.

When computing the discrete logarithms of small elements, we do not need to compute them directly modulo $q - 1$. It is sufficient to compute logarithms modulo primes $l \mid q-1$ and then use Chinese Reminder Theorem to obtain logarithms modulo $q-1$.

Having enough relations with ideals transfered to elements, we can create the matrix, solve it modulo l and we get the discrete logarithms of the elements from the factor basis. At the end, having the discrete logarithms of the elements, we can finally apply maps ϕ_i and take logarithms of both sides to get

$$\log(\phi_i(a+b\theta_i)) \equiv \log(\phi_i(u)) + \sum_j e_j \log(\phi_i(\gamma_j)) \bmod l, \tag{3.8}$$

which yields

$$\log(a+bm) \equiv \log(\bar{u}) + \sum_j e_j \log(\bar{\gamma_j}) \bmod l, \tag{3.9}$$

where for any element $x \in \mathcal{O}_{K_i}$ we define $\bar{x} = \phi_i(x)$.

Once having discrete logarithms of elements from the factor basis modulo $q-1$, we can used them to compute a searched discrete logarithm. When trying to compute $t^x \equiv u \pmod q$, the method is as follows: choose a random $s \in \mathbb{Z}$ and compute $t^s \cdot u$. Verify that all the divisors of $t^s \cdot u \pmod q$ belong to the factor basis, that means that we have already obtained the discrete logarithms of them. If it does not hold, choose another s. Once s is found, we get

$$t^s \cdot u = \prod_{i=1}^{r} c_i^{d_i} \pmod q.$$

Taking logarithm, we obtain

$$\log_t(t^s \cdot u) = \log_t(t^s \cdot t^x) \equiv \sum_{i=1}^{r} d_i \log_t(c_i) \pmod q,$$

then

$$s + x \equiv \sum_{i=1}^{r} d_i \log_t(c_i) \pmod q.$$

As we already know the value of the discrete logarithms $\log_t(c_i)$, we can now easily compute x, which is our searched discrete logarithm.

3.2 Choice of Polynomials

The choice of the polynomials is an important aspect of the `NFS` algorithm. When good polynomials are chosen, norm of the elements in the number fields is of reasonable size, more smooth elements are found and therefore smaller values for smoothness bound and sieve limit are needed. In general, we need to find two irreducible polynomials f_1 and f_2 with a common root modulo p, denoted by m. For the algorithm to be efficient, these polynomials should have small coefficients. There are several ways how to determine good polynomials and we present three of them.

Base m Method

The most common technique for building polynomials for the number field sieve in $\mathbb{F}_p$ is known as the Base m method. One simply chooses a number m and writes p in base m as $\sum a_i m^i$. Clearly, $X - m$ and $\sum a_i X^i$ are two polynomials with the same root m. In this case, the first polynomial is of degree one and therefore it corresponds to the field $\mathbb{Q}$. The second one, $f = \sum a_i X^i$, forms a number field $K = \mathbb{Q}[\theta]$, where $\theta \in \mathbb{C}$ satisfies $f(\theta) = 0$. Schirokauer in his work [26] mentioned, that if m is chosen as $\lfloor p^{1-d} \rfloor$, then the coefficients of f are bounded by $p^{1/d}$, f has a root modulo p of size at most $p^{1/d}$ and f is monic. The first and the second properties are crucial to the running time of the algorithm as they influence the smoothness bound and the sieve limit.

Lattice Reduction

Another technique in $\mathbb{F}_p$ producing two polynomials with degrees d and $d+1$ was introduced by Joux and Lercier in [25]. It is in fact a generalization of the Gaussian integer method. In the beginning, we choose a polynomial of a degree $d+1$ with small coefficients and with m as a root modulo p. Then, we need to find a second polynomial of degree d also having root m. According Joux and Lercier, this can be done by reducing the following lattice:

$$\begin{pmatrix} D & D(m \bmod p) & D(m^2 \bmod p) & \cdots & D(m^d \bmod p) & Dp \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix},$$

where columns contain basis vectors and D is an arbitrarily chosen constant. They also suggest to reduce the lattice by subtracting multiples of the first columns from the others and by removing the first line and column. By doing these and moving the last column to the first position, we get:

$$\begin{pmatrix} p & -m & -m^2 & \cdots & -m^d \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Now, when `LLL` algorithm is used for the lattice reduction, the first vector of the reduced lattice would be of the form

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix}.$$

This gives the coefficients of the second polynomial $\sum_{i=0}^{d} a_i X^i$, where $a_0, a_1, \ldots, a_d$ are bounded by $p^{1/(d+1)}$. In the case $d = 1$, it is equivalent to the Gaussian integer method. When $d = 2$, two polynomials with degree 2 and 3 are obtained, which yield two distinct number fields.

Method +p

In case of choosing both polynomials with the same degree, there is a very easy way how to choose the second polynomial presented by Joux et al. in [27]. First, choose a monic and irreducible polynomial f_1 of degree d with small coefficients. Then authors simply set the second polynomial f_2 to be equal to the polynomial $f_1 + p$. Since $f_2 \equiv f_1 \pmod{p}$, both polynomials have the same root in $\mathbb{F}_{p^n}$. This method is sufficient in lot of cases and we adopt the same principle in our examples.

3.3 Factoring Ideals in $\mathcal{O}_K$

In this section, we present how we factor ideals in the ring of integers. As $\mathcal{O}_K$ is the Dedekind domain, it ensures the unique factorization of the ideals to prime ideals.

In the algorithm, we look for coprime pairs of integers (a, b) such that the principal ideal $(a - b\theta_i)$ factors into prime ideals of a small norm. Prime ideals that might occur in such a factorization are given by the Lemma 10.5.1 in [7] and is as follows:

Lemma 3.1. *Let $K = \mathbb{Q}[\theta]$ and (a, b) be coprime integers. Then any prime ideal P which divides $(a - b\theta)$, either divides the index $f_\theta = [\mathcal{O}_K : \mathbb{Z}[\theta]]$ or is of degree one.*

Therefore, if we denote by F the set of degree one prime ideals of norm smaller than B and those finitely many prime ideals that divide the index by f_θ then we try to factorize ideals $(a - b\theta)$ over F. As the authors are mentioning in [27], each prime ideal of degree one is generated by $(p_i, \theta - c_{p_i})$ with p_i a rational prime smaller than B and c_{p_i} a root of $f(x)$ modulo p_i.

Furthermore, the decomposition of the ideal $(a - b\theta)$ into prime ideals is not difficult. The norm of the ideal $N_{K/\mathbb{Q}}(a - b\theta) = b^{\deg(f)} f(a/b)$ is computed and then tested whether it is B-smooth. If so, we can write $N_{K/\mathbb{Q}}(a - b\theta) = \sum_i p_i^{e_i}$. For p_i, we distinguish two possibilities:

- For rational primes $p_i \nmid f_\theta$, we get $P_i = (p_i, \theta - c_{p_i})$ with $c_{p_i} \equiv a/b \pmod{p_i}$ occurs in the ideal factorization of $(a - b\theta)$ and P_i-adic valuation is precisely e_i.
- For rational primes $p_i \mid f_\theta$, it is more complicated and one can use Algorithm 4.8.17 in [7].

At the end, from the pairs of coprime integers (a, b) we obtain ideal decompositions

$$(a - b\theta) = \prod_i P_i^{e_i}. \tag{3.10}$$

3.4 From Ideals to Elements

While using the number field sieve algorithm, one ends up with computed discrete logarithms of ideals. To compute the searched discrete logarithm, we need to transform these ideals to the elements. To achieve this, we transform the relations between ideals to multiplicative relations including elements only. After applying the maps ϕ_i and taking logarithms, we obtain linear equations between logarithms of elements in $\mathbb{F}_q$ modulo the prime factor l.

We analyze the case, when the class number of a number field is one and has a computable unit group.

K with Class Number One and Computable Unit Group

As it was already mentioned in the description of the algorithm, when the class number of a number field K equals one, every ideal $I \in \mathcal{O}_K$ is principal, moreover, there exists $\gamma_I \in \mathcal{O}_K$, such that $I = (\gamma_I)$. Then the ideal decomposition $(a - b\theta) = \prod_i P_i^{e_i}$ leads to

$$a - b\theta = u \cdot \prod_i \gamma_i^{e_i} \tag{3.11}$$

with $(\gamma_j) = P_j$ and u a unit in $\mathcal{O}_K$. Denote by (r_1, r_2) the signature of K, then the unit group $\mathcal{O}_K^* \cong \mu(K) \times \mathbb{Z}^{r_1+r_2-1}$, with $\mu(K) = \langle u_0 \rangle$ is a finite cyclic group of order v. Joux et al. in [27] further assume, that we can compute a system of fundamental units $u_1, u_2, \ldots, u_r$ with $r = r_1 + r_2 - 1$, so then we can write $u = u_0^{n_0} \cdot u_1^{n_1} \cdot \ldots \cdot u_r^{n_r}$. Thus we obtain r logarithmic maps λ_i for $i = 1, \ldots, r$ defined by

$$\lambda_i : \mathcal{O}_K^* \to \mathbb{Z} : u \mapsto n_i, \tag{3.12}$$

and a logarithmic map $\lambda_0 : \mathcal{O}_K^* \mapsto \mathbb{Z}/v\mathbb{Z} : u \mapsto n_0$. Ultimately, we obtain the decomposition

$$a - b\theta = \prod_{i=0}^{r} u_i^{\lambda_i(u)} \prod_i \gamma_i^{e_i}. \tag{3.13}$$

Now, by applying ϕ_i and taking the logarithms of both sides we get

$$\log_t(a - b\theta) \equiv \sum_{i=0}^{r} \lambda_i(u) \log_t \bar{u}_i + \sum_i e_i \log_t \bar{\gamma}_i \bmod l, \tag{3.14}$$

where, for any element $x \in \mathcal{O}_{K_i}$ we define $\bar{x} = \phi_i(x)$.

3.5 Example of Number Field Sieve Algorithm for Discrete Logarithm in $\mathbb{Z}_p$

In this section, we show an example of the number field sieve algorithm for the basic case $\mathbb{F}_p$ considering p as a medium size prime.

Let us choose $p = 11$. Our group is then $\mathbb{F} = \mathbb{Z}_{11}$ and we want to find

$$\log_t(u) \equiv y \pmod{p-1},$$

for some u. The element t needs to be a generator of the group $\mathbb{F}^*$. In our case, the generator of $\mathbb{Z}_{11}^*$ is 2, therefore $t = 2$.

The next step is to choose two monic irreducible polynomials, e.g.

$$f_1 = x^2 + 7 \quad \text{and} \quad f_2 = x - 2.$$

Both polynomials have the same root modulo 11, namely $m = 2$. The first polynomial f_1 determines a number field $K_1 = \mathbb{Q}[\sqrt{7}i]$. The second polynomial f_2 yields $K_2 = \mathbb{Q}$. The problem is that we need to work in an integral closure and $\mathbb{Q}[\sqrt{7}i]$ is not integrally closed. Therefore, we need to find the extension of the $\mathbb{Q}[\sqrt{7}i]$ that is integrally closed. For example, we can take $\mathbb{Q}[\frac{1+\sqrt{7}i}{2}]$.

Moreover, we define homomorphisms that allow us to transform elements from the number fields back to $\mathbb{Z}_{11}$ by using the common root of polynomials f_1 and f_2. The homomorphisms are defined as

$$\begin{aligned} &\phi_1 : a + b\sqrt{7}i \mapsto a + 2b \pmod{11} \quad \text{and} \\ &\phi_2 : a \mapsto a \pmod{11}, \end{aligned}$$

so we have $\theta_1 = \sqrt{7}i$ and $\theta_2 = 2$.

In the next step let us choose a factorization basis for K_1 as the set $\{-1, \sqrt{7}i, \frac{\sqrt{7}i+1}{2}, \frac{\sqrt{7}i-1}{2}\}$ composed of the generators of the prime ideals and some units and for K_2 the set $\{-1, 2, 3, 5\}$ composed of the natural numbers and -1. In the sieving stage, we are looking for pairs (a, b), $a, b \in \mathbb{Z}$, such that $\gcd(a, b) = 1$ and the ideal $(a + b\theta_i)$ factorizes to the prime ideals with generators contained in the basis.

One such a pair is $(-1, 1)$ as $(\theta_1 - 1)$ factorizes to $-1 \cdot (\frac{\sqrt{7}i+1}{2})^1 \cdot (\frac{\sqrt{7}i-1}{2})^2$ in K_1 and $(\theta_2 - 1)$ factorizes to $-1 \cdot 2 \cdot 5$ in K_2.

More of these pairs can be found in the forthcoming table where rows correspond to the relations. The elements characterizing the columns are the generators of the ideals to which $(a + b\theta_i)$ factorizes, i.e., for K_1 they are the generators of the ideals from the factorization basis. The numbers in the table determine exponents of the factorization in K_1 and K_2.

	-1	$\sqrt{7}i$	$\frac{\sqrt{7}i+1}{2}$	$\frac{\sqrt{7}i-1}{2}$	-1	2	3	5
$\theta_i + 1$	1	0	2	1	0	0	1	0
$\theta_i - 1$	1	0	1	2	0	0	0	0
$\theta_i + 3$	0	0	1	3	0	0	0	1
$\theta_i - 3$	1	0	3	1	1	0	0	0
$\theta_i + 5$	0	0	4	1	0	0	0	0
$\theta_i - 5$	0	0	1	4	1	0	1	0
$\theta_i + 7$	0	1	1	2	0	0	2	0
2	0	0	0	0	0	1	0	0

To get a non-homogeneous matrix we add one more relation $x^1 \equiv 2 \pmod{10}$, which then yields the unique solution of the matrix. So the matrix M composed of the relations above is as follows

$$M = \begin{pmatrix} 1 & 0 & 2 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 0 & -1 \\ 1 & 0 & 3 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

with vector $b = [0, 0, 0, 0, 0, 0, 0, 1]^\top$ on the right side. Let M1 $= M^\top$ and by using the PARI/GP command `matsolvemod(M1*M,10,M1*b)` we get the solution $[-5, 1, -3, -1, -5, 1, -2, 4]^\top$, as

$\log_2(\phi_1(-1)) = \log_2(10) = 5 \equiv -5 \pmod{10}$,
$\log_2(\phi_1(\sqrt{7}i)) = \log_2(2) \equiv 1 \pmod{10}$,
$\log_2(\phi_1(\frac{\sqrt{7}i+1}{2})) = \log_2(1/2 + 1) = \log_2(7) \equiv -3 \pmod{10}$,
$\log_2(\phi_1(\frac{\sqrt{7}i-1}{2})) = \log_2(-1/2 + 1) = \log_2(-5) = \log_2(6) = 9 \equiv -1 \pmod{10}$,
$\log_2(-1) = \log_2(10) = 5 \equiv -5 \pmod{10}$,
$\log_2(2) \equiv 1 \pmod{10}$,
$\log_2(3) = 8 \equiv -2 \pmod{10}$,
$\log_2(5) = 4 \pmod{10}$.

So the computed logarithms are (mod 10):

$\log_2(-1) = 5,$	$\log_2(3) = 8,$	$\log_2(6) = 9,$	$\log_2(7) = 7,$
$\log_2(2) = 1,$	$\log_2(5) = 4,$	$\log_2(10) = 5.$	

Now we are able to compute the discrete logarithm of our chosen number from the group $\mathbb{Z}_{11}$. For some numbers, we already have the discrete logarithms computed as we chose small numbers, but let us compute the discrete logarithm of 9, i.e.

$$y = \log_{11}(9) \pmod{10}.$$

We want to use the precomputed discrete logarithms of the elements from the basis. By using the same method as in the Index Calculus algorithm we choose some power of 2, multiply with 9, factorize and verify that all divisors have precomputed discrete logarithm.

If the exponent is 1, then $9 \cdot 2^1 \equiv 7 \pmod{11}$. We know that $\log_2 7 \equiv 7 \pmod{10}$, therefore

$$\log_2(9 \cdot 2^1) \pmod{10} = \log_2 2^{y+1} \pmod{10} = \log_2 7 \pmod{10}.$$

Implying

$$y + 1 \equiv 7 \pmod{10}, \text{ which means that } y \equiv 6 \pmod{10}.$$

So $2^6 \equiv 9 \pmod{11}$.

We showed a basic example of the number field sieve algorithm for the discrete logarithm problem using only one number field. The next section covers a more generic case using two number fields.

3.6 Example of Number Field Sieve Algorithm for Discrete Logarithm in $\mathbb{Z}_{p^n}$

In this section, we demonstrate the number field sieve algorithm when computing a discrete logarithm in a group $\mathbb{Z}_{p^n}$. We use two number fields and we consider p as a medium size prime.

Let us take $p = 5$ and $n = 2$. We want to compute a discrete logarithm in the group $\mathbb{Z}_{5^2}$. Generator of $\mathbb{Z}^*_{5^2}$ is 2 and the order of the group $\operatorname{ord}(\mathbb{Z}^*_{5^2}) = 20$. Therefore we want to compute

$$\log_2(u) = y \pmod{20}$$

for some chosen u.

As $\mathbb{Z}_{5^2}$ is not a field, we need to choose two polynomials with the same root modulo 5^2.

Our chosen polynomials are

$$f_1 = x^2 + 19 \quad \text{and} \quad f_2 = x^2 + 19 - p^2 = x^2 - 6.$$

Both the polynomials have modulo 25 the same root 9.

In the next step, we define number fields generated by these polynomials as follows

$$K_1 = \mathbb{Q}[\sqrt{19}i] \quad \text{and} \quad K_2 = \mathbb{Q}[\sqrt{6}],$$

so we have $\theta_1 = \sqrt{19}i$ and $\theta_2 = \sqrt{6}$. The class number of both number fields is 1, therefore their integral closures are principal ideal domains and we can transform the ideals to the elements.

Moreover, we need homomorphisms that map our number fields K_1 and K_2 to $\mathbb{Z}_{5^2}$. Again, we use the common root of the polynomials f_1 and f_2. Let

$$\phi_1 : a + b\sqrt{19}i \mapsto a + 9b \pmod{25} \quad \text{and}$$
$$\phi_2 : a + b\sqrt{6} \mapsto a + 9b \pmod{25}.$$

Now, we define bases for both number fields and we look for relations between these elements. We take such elements that their norm in the number field is a prime number or a power of a prime number and avoid elements which images in $\mathbb{Z}^*_{5^2}$ equal 0 as we assume that all images by ϕ_1 and ϕ_2 of elements from the basis belong to $\mathbb{Z}^*_{5^2}$. Such element, whose image in $\mathbb{Z}^*_{5^2}$ is 0, is in our case $\frac{-\theta_1-1}{2}$ in K_1 and $-\theta_2 - 1$ in K_2.

Let's take basis in K_1 as $\{-1, 2, \frac{\theta_1-1}{2}, \frac{\theta_1-3}{2}, \frac{-\theta_1-3}{2}\}$ and basis in K_2 as $\{-1, -\theta_2 - 2, \theta_2 + 3, \theta_2 - 3, -\theta_2 + 1\}$. Elements of the basis are the generators characterizing the ideals. Some of the discovered relations are shown in the table:

	-1	2	$\frac{\theta_1-1}{2}$	$\frac{\theta_1-3}{2}$	$\frac{-\theta_1-3}{2}$	-1	$-\theta_2-2$	θ_2+3	θ_2-3	$-\theta_2+1$
$\theta_i - 3$	0	1	0	1	0	0	0	0	1	0
$8\theta_i - 3$	0	0	2	2	0	1	0	1	0	3
$3\theta_i + 2$	0	0	2	0	1	1	1	0	0	2
$\theta_i + 4$	1	0	1	1	0	0	1	0	0	1
$\theta_i + 9$	1	1	2	0	0	0	0	1	0	2
$\theta_i - 1$	0	1	1	0	0	1	0	0	0	1
$\theta_i + 3$	1	1	0	0	1	0	0	1	0	0
$2\theta_i - 7$	1	0	3	0	0	1	0	0	0	2

When enough such relations are found and the matrix is solved, that we get discrete logarithms of elements from the basis. Using homomorphisms ϕ_1 and ϕ_2 we convert logarithms of ideals to the logarithms of elements from $\mathbb{Z}^*_{5^2}$.

$\log_2 \phi_1(-1) = \log_2(-1) \equiv 10 \pmod{20}$
$\log_2 \phi_1(2) = \log_2(2) \equiv 1 \pmod{20}$
$\log_2 \phi_1(\frac{\theta_1-1}{2}) = \log_2(-1/2+9/2) = \log_2(-1\cdot 13+9\cdot 13) = \log_2(104) \equiv \log_2(4) \equiv 2 \pmod{20}$
$\log_2 \phi_1(\frac{\theta_1-3}{2}) = \log_2(9\cdot 13 - 3\cdot 13) \equiv \log_2(3) \equiv 7 \pmod{20}$
$\log_2 \phi_1(\frac{-\theta_1-3}{2}) = \log_2(-9\cdot 13 - 3\cdot 13) \equiv \log_2(19) \equiv 18 \pmod{20}$

$\log_2 \phi_2(-1) = \log_2(-1) \equiv 10 \pmod{20}$
$\log_2 \phi_2(-\theta_2 - 2) = \log_2(-11) \equiv \log_2(14) \equiv 6 \pmod{20}$
$\log_2 \phi_2(\theta_2 + 3) = \log_2(12) \equiv 9 \pmod{20}$
$\log_2 \phi_2(\theta_2 - 3) = \log_2(6) \equiv 8 \pmod{20}$
$\log_2 \phi_2(-\theta_2 + 1) = \log_2(-8) \equiv 13 \pmod{20}$

So the computed logarithms are (mod 20):

$$\begin{array}{lll} \log_2(-1) = 10, & \log_2(4) = 2, & \log_2(14) = 6, \\ \log_2(2) = 1, & \log_2(6) = 8, & \log_2(17) = 13, \\ \log_2(3) = 7, & \log_2(12) = 9. & \log_2(19) = 18, \end{array}$$

and from these we can compute any discrete logarithm from group $\mathbb{Z}^*_{5^2}$.

For example, let us compute x such that

$$2^x \equiv 22 \pmod{25}.$$

That is $\log_2(22) \equiv x \pmod{20}$. We find a power of 2, such that $22 \cdot 2^y \pmod{25}$ factorizes to the numbers for which we have already computed discrete logarithms. One such power is 11 as

$$22 \cdot 2^{11} \pmod{25} = 6 = 2 \cdot 3.$$

Then it holds

$$22 \cdot 2^{11} = 2^{x+11} \equiv 2 \cdot 3 \pmod{25},$$

and by applying logarithms on both sides we get

$$\log_2(2^{x+11}) \equiv \log_2(2) + \log_2(3) \pmod{20}.$$

We use the computed values for $\log_2(2)$ and $\log_2(3)$ (mod 20) and get

$$x = 1 + 7 - 11 = -3 \equiv 17 \pmod{20}.$$

So $x \equiv 17 \pmod{20}$, which is the value we were looking for.
Consequently, $2^{17} \equiv 22 \pmod{25}$.

Chapter 4

Number Field Sieve Algorithm for Factorization

In the previous chapter we analyzed the number field sieve algorithm for discrete logarithm. As we want to compare the number field sieve for discrete logarithm and for factorization, we devote this chapter to the `NFS` for factorization and provide the comparison in the following chapter.

Similarly to the discrete logarithm, the number field sieve is currently the asymptotically fastest known algorithm for factoring integers and it is also based on the Index Calculus algorithm. There are two forms of the number field sieve algorithm for factorization. They can be distinguished by whether N itself has a special simple form, then the algorithm is called the Special `NFS` or `SNFS`, or whether number N is a general integer, then the algorithm is called the General `NFS`, or `GNFS`. We cover the general case and present only a brief description of the number field sieve algorithm for factorization.

As it was mentioned, the number field sieve algorithm for factorization is motivated by Kraitchik's method. When having a number N to factorize and pairs x_i, y_i, such that

$$x_i^2 \equiv y_i^2 \bmod N, \tag{4.1}$$

then $\gcd(N, x_i - y_i)$ is a non-trivial divisor of N with probability $1/2$. After trying enough pairs, N would be factorized.

In the next section, we define number field sieve algorithm for factorization in more detail.

4.1 Number Field Sieve Algorithm

Let N be a number which we want to factorize. The number field sieve algorithm requires two distinct irreducible polynomials

$$f_i(x) = \textstyle\sum_{j=0}^{d_i} f_{i,j} x^j \qquad \text{for } i = 1, 2,$$

of degree d_i, respectively, with $f_{i,j} \in \mathbb{Z}$, such that there exists a common root m modulo N:

$$f_i(m) \equiv 0 \pmod{N}.$$

In some cases, the second polynomial is chosen to be of degree one and then only one number field is used. We keep the notation with two number fields in our work which is the generic case. To choose the polynomial, one can use one of the three methods described in Section 3.2. Let θ_i be a complex root of the polynomial f_i, then number fields are defined as $K_i = \mathbb{Q}[\theta_i]$ and contain rings of integers $\mathcal{O}_{K_i}$. Since m is a root of $f_i \pmod{N}$, there exist natural homomorphisms $\phi_i : \mathbb{Q}[\theta_i] \to \mathbb{Z}/N\mathbb{Z}$ defined by $\theta_i \mapsto m \pmod{N}$ and $\phi_i(a) \equiv a \pmod{N}$ for $a \in \mathbb{Z}$.

In NFS for discrete logarithm, the main principle is to compute the discrete logarithms of small elements from some basis and to use them to compute the searched discrete logarithm. In NFS for factorization, we want to find $\gamma_1 \in \mathcal{O}_{K_1}$ and $\gamma_2 \in \mathcal{O}_{K_2}$ such that

$$\phi_1(\gamma_1^2) \equiv \phi_2(\gamma_2^2) \pmod{N},$$

since then $x = \phi_1(\gamma_1)$ and $y = \phi_2(\gamma_2)$ satisfy

$$x^2 \equiv y^2 \pmod{N}$$

and $\gcd(x - y,\, N)$ may disclose a factor of N. This is achieved by looking for coprime integer pairs (a, b) such that the products

$$\gamma_i^2 = \prod (a - b\theta_i) \quad \text{for } i = 1, 2,$$

produce a square in their respective ring $\mathcal{O}_{K_i}$. Then the images of both products are congruent modulo N as

$$\phi_1(a - b\theta_1) \equiv a - bm \equiv \phi_2(a - b\theta_2) \pmod{N}.$$

Similarly to the NFS for the discrete logarithm, we choose a smoothness bound B_i and require both $a - b\theta_i$ to be B_i-smooth. Since the factorization in the number fields used in NFS is not unique in general, we would not be working with the elements $a - b\theta_i$, but with the ideals $(a - b\theta_i)$. Factorization of the ideals is done in the ring of integers $\mathcal{O}_{K_i}$ and, as it was mentioned, $\mathcal{O}_{K_i}$ is a Dedekind domain ensuring a unique factorization. To verify smoothness of the ideal, one can use Lemma 1.42 or the link between the norm of the elements and the norm of the ideals. If the norm of $a - b\theta_i$ factors into primes not exceeding a smoothness bound B_i, then the norm of the prime ideals where ideal $(a - b\theta_i)$ factorizes to is also less than the B_i. The norm of the $a - b\theta_i$ is expressed by

$$N(a - b\theta_i) = F_i(a, b),$$

where

$$F_i(a, b) = f_i\left(\frac{a}{b}\right) b^{d_i}$$

is the homogeneous form of the polynomials f_i.

If we find enough relations coming from the pairs (a_j, b_j), we can choose subset of indices L such that

$$\prod_{j \in L} ((a_j + b_j\theta_1)\mathcal{O}_{K_1}) = P_1^{2e_1} \cdot P_2^{2e_2} \cdot \ldots \cdot P_{k_1}^{2e_{k_1}} = \gamma_1^2, \tag{4.2}$$

$$\prod_{j \in L} ((a_j + b_j\theta_2)\mathcal{O}_{K_2}) = Q_1^{2e_1} \cdot Q_2^{2e_2} \cdot \ldots \cdot Q_{k_2}^{2e_{k_2}} = \gamma_2^2 \tag{4.3}$$

Now, we need to find the square roots of γ_1^2 and γ_2^2. This is considered to be difficult. Because even if we know

$$\prod_{j \in L}((a_j + b_j\theta_i)\mathcal{O}_{K_i}) = \gamma_i^2 \quad i = 1, 2 \tag{4.4}$$

for some $\gamma_i \in \mathcal{O}_{K_i}$, the ideal γ_i need not be principal. Furthermore if we would find the aforementioned γ_i principal, i.e. $\gamma_i = \alpha_i \mathcal{O}_{K_i}$, then it is not necessarily true that $\prod_{j \in L}(a_j + b_j\theta_i) = \alpha_i^2$. The solution to this problem is to use the quadratic characters. It is true that $\prod_{j \in L}(a_j + b_j\theta_i) \in N_i = \{\delta \in K_i^*; \delta\mathcal{O}_{K_i} = A^2, A \text{ is a fractional ideal}\}$. If we can prove that $\prod_{j \in L}(a_j + b_j\theta_i) \in (K_i^*)^2$, then $\prod_{j \in L}(a_j + b_j\theta_i) = \alpha_i^2$ and the ideals γ_i would be principal. Moreover it ensures the existence of the α_i. We describe how to use quadratic characters and how to choose the subset of indices L from all the pairs (a_j, b_j).

Let us have a factorization of ideals $\{(a_1 + b_1\theta_i), \ldots, (a_r + b_r\theta_i)\}$ as follows

$$\begin{aligned}
(a_1 + b_1\theta_1) = &\ P_1^{e_{11}} \cdot P_2^{e_{12}} \cdot \ldots \cdot P_{k_1}^{e_{1k_1}} \\
\vdots \qquad & \qquad\qquad \vdots \\
(a_r + b_r\theta_1) = &\ P_1^{e_{r1}} \cdot P_2^{e_{r2}} \cdot \ldots \cdot P_{k_1}^{e_{rk_1}} \\
(a_1 + b_1\theta_2) = &\ Q_1^{f_{11}} \cdot Q_2^{f_{12}} \cdot \ldots \cdot Q_{k_1}^{f_{1k_2}} \\
\vdots \qquad & \qquad\qquad \vdots \\
(a_r + b_r\theta_2) = &\ Q_1^{f_{r1}} \cdot Q_2^{f_{r2}} \cdot \ldots \cdot Q_{k_1}^{f_{rk_2}}
\end{aligned}$$

Denote by $\{U_{i1}, \ldots, U_{il_i}\}$ the prime ideals in $\mathcal{O}_{K_i}$ over non-special prime numbers larger than the boundary for factorization basis. These prime ideals are used as quadratic characters. Elements $a_j + b_j\theta_i$ do not lie in this set of prime ideals therefore we never get $(\frac{a_j + b_j\theta_i}{U_{ik}}) = 0$. We define

$$g_{jk} = \begin{cases} 1 & \text{if } \left(\dfrac{a_j + b_j\theta_1}{U_{1k}}\right) = -1, \\ 0 & \text{otherwise} \end{cases}$$

and

$$h_{jk} = \begin{cases} 1 & \text{if } \left(\dfrac{a_j + b_j\theta_2}{U_{2k}}\right) = -1, \\ 0 & \text{otherwise} \end{cases}$$

for $j = 1, \ldots r$, $k = 1, \ldots, l_i$. Then we compose the matrix M as follows

$$M^T = \begin{pmatrix} e_{11} & \cdots & e_{1k_1} & f_{11} & \cdots & f_{1k_2} & g_{11} & \cdots & g_{1l_1} & h_{11} & \cdots & h_{1l_2} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ e_{r1} & \cdots & e_{rk_1} & f_{r1} & \cdots & f_{rk_2} & g_{r1} & \cdots & g_{rl_1} & h_{r1} & \cdots & h_{rl_2} \end{pmatrix}.$$

One column of the matrix M is composed of powers of the prime ideals from the factorization of ideals $(a_j + b_j\theta_1)$ in $\mathcal{O}_{K_1}$ and $(a_j + b_j\theta_2)$ in $\mathcal{O}_{K_2}$, then from quadratic characters of the elements $a_j + b_j\theta_1$ in $\mathcal{O}_{K_1}$ and quadratic characters of the elements $a_j + b_j\theta_2$ in $\mathcal{O}_{K_2}$.

In reality, the matrix M is very big, therefore some special algorithms are used to solve such matrices. As it was already mentioned, one of them is Wiedemanns algorithm presented in [9].

Each solution of the matrix M over $\mathbb{Z}_2$ determines the index of the set L such that

$$\prod_{j\in L}(a_j + b_j\theta_1)\mathbb{Z}[\theta_1] = \prod_{j=1}^{k_1} P_j^{2e_j},$$

$$\prod_{j\in L}(a_j + b_j\theta_2)\mathbb{Z}[\theta_2] = \prod_{j=1}^{k_2} Q_j^{2f_j}.$$

The existence of the square root is guaranteed by the quadratic characters. Moreover, we need the square root of $\prod_{j\in L}(a_j + b_j\theta_i) \in \mathbb{Z}[\theta_i]$ to belong to $\mathbb{Z}[\theta_i]$. This is achieved by multiplying the product by $(f_i'(\theta_i))^2$. Then we have

$$(f_2'(m))^2 \cdot \phi_1((f_1'(\theta_1))^2 \prod_{j\in L}(a_j+b_j\theta_1)) \equiv (f_1'(m))^2 \cdot \phi_2((f_2'(\theta_2))^2 \prod_{j\in L}(a_j+b_j\theta_2)) \bmod N.$$

Now, we need to find the square root of

$$\alpha_i = \sqrt{(f_i'(\theta_i))^2 \cdot \prod_{j\in L}(a_j + b_j\theta_i)}. \tag{4.5}$$

Special methods were developed to compute this. First, the Newton's method described by Lenstra and Lenstra in [19], which requires to work with very large numbers. A second method comes from Jean-Marc Couveignes defined in [28] and it requires an odd degree of the minimal polynomial. And another method defined by Montgomery in [29] with different approach using known factorization of the ideals. Using one of these methods, we get

$$(f_2'(m))^2 \cdot \phi_1(\alpha_1^2) \equiv (f_1'(m))^2 \cdot \phi_2(\alpha_2^2) \bmod N, \tag{4.6}$$

which is

$$(f_2'(m) \cdot \phi_1(\alpha_1))^2 \equiv (f_1'(m) \cdot \phi_2(\alpha_2))^2 \bmod N. \tag{4.7}$$

If we denote $x = f_2'(m)\phi_1(\alpha_1)$ and $y = f_1'(m)\phi_2(\alpha_2)$, then $\gcd(N, x - y)$ is a non-trivial factor of N with probability $1/2$.

Chapter 5

Comparison of Number Field Sieve Algorithm for Factorization and Discrete Logarithm

The main goal of this work is to investigate the number field sieve algorithm for discrete logarithm and to compare the NFS algorithm for factorization and for discrete logarithm. In the previous chapters, we covered the analysis of the NFS for discrete logarithm and the description of NFS for factorization. This chapter covers the comparison of these two algorithms.

Even though both algorithms are based on the same idea, many parts are different. The algorithms solve distinct mathematical problems, that is where the major difference comes from. The parts of both algorithms which can be considered the same are the sieving stage, the choice of polynomials and the usage of the integral closure. More interesting is the list of differences. When we look closer, whole sections are different. We can list them as follows:

- The reason of finding the pairs (a, b).
- The creation of the matrix M – adding quadratic characters in the factorization and units in the discrete logarithm.
- The solution of the matrix M – in the factorization we just get indices of pairs that create the squares, in the discrete logarithm we get directly the discrete logarithms of the elements from the factor bases.
- Final step - finding the searched discrete logarithm or the divisor of a number N.

5.1 Choice of Polynomials and Sieving

These are the only sections that are in principle the same in both algorithms. In the choice of the polynomials we are looking for good polynomials that generate the number fields where we can find enough B-smooth relations. The covered techniques, base m method, lattice reduction and method $+p$, are used to find these polynomials in both algorithms. It depends only on the independent settings of each case if one requires two polynomials of the same degree or polynomials of degrees different by one, or just to use one number field and therefore having one polynomial of degree one.

In the sieving stage of both algorithms, we are looking for the pairs (a, b) such that ideals $(a + b\theta)$ are B-smooth. Since we are factoring ideals to prime ideals in both cases, this does not differ.

5.2 Creation and Solution of Matrix M

Sieving steps itself does not differ in the factorization and in the discrete logarithm variants of the number field sieve algorithm. However, the difference is found in the reason of creating the matrix from the relations. In the `NFS` for factorization, we want to find a subset L of all the relations such that they create the squares:

$$\prod_{j\in L}((a_j + b_j\theta_1)\mathcal{O}_{K_1}) = P_1^{2e_1} \cdot P_2^{2e_2} \cdot \ldots \cdot P_{k_1}^{2e_{k_1}} = \gamma_1^2,$$
$$\prod_{j\in L}((a_j + b_j\theta_2)\mathcal{O}_{K_2}) = Q_1^{2e_1} \cdot Q_2^{2e_2} \cdot \ldots \cdot Q_{k_2}^{2e_{k_2}} = \gamma_2^2.$$

Therefore, we compose the matrix M in a way that it gives us this solution.

In the `NFS-DLP`, we do not need this condition. We want to compute the discrete logarithms of the elements from the basis, so we just need to find enough relations.

When enough relations are found, the matrix M is composed. In the case of the factorization, more information is needed to be able to find the square roots. All the required values for the quadratic characters have to be added to the matrix, which makes the matrix huge but also sparse. Other difference is that in the factorization the matrix can be composed in $\mathbb{Z}_2$. As it was mentioned in Section 4.1, the columns representing quadratic characters contain only values 0 and 1. And as we need to know just when we get the squares, it is enough to know which powers are coming in the even or odd number.

We have different situation in the discrete logarithm case. As we do not need any square roots, we do not use the quadratic characters. On the other hand, when dealing with the number fields of the class number one, we need to add the units to the relations when considering them as the algebraic integers. We factor the ideals to the prime ideals, transform them to the algebraic integers, compute the units and create the matrix. By solving the matrix in $\mathbb{Z}$, we directly get the discrete logarithms of the generators of the ideals. We cannot solve the matrix in $\mathbb{Z}_2$ in this case.

5.3 Finding the Searched Discrete Logarithm or the Divisor of a Number N

The final step of the algorithms differs a lot as we are solving different mathematical problems. In the factorization, we want to find a divisor of our chosen number N. After solving the matrix M, we get indices of the set L to determine which pairs (a_j, b_j) we need to use to get

$$\prod_{j\in L}(a_j + b_j\theta_1)\mathbb{Z}[\theta_1] = \prod_{j=1}^{k_1} P_j^{2e_j},$$

$$\prod_{j\in L}(a_j + b_j\theta_2)\mathbb{Z}[\theta_2] = \prod_{j=1}^{k_2} Q_j^{2f_j}.$$

As it was mentioned before, the existence of the square root is guaranteed by the quadratic characters. Two more steps are needed in order to determine whether we can find a factor of a number N. First, multiplying the product by $(f_i'(\theta_i))^2$ to achieve that the square roots of $\prod_{j\in L}(a_j + b_j\theta_i)$ belong to $\mathbb{Z}[\theta_i]$. Second, compute the square root. Once this is all done, we obtain a value for x and y computed in the previous chapter, such that $x \equiv y \pmod N$; therefore, we can calculate $\gcd(x - y, N)$ and verify whether it is a divisor of the number N.

In the `NFS-DLP` case, we finish with calculated discrete logarithms of the elements from our factor bases. To compute our searched discrete logarithm x such that $t^x \equiv u \pmod q$, we need to randomly choose $s \in \mathbb{Z}$ and compute $t^s \cdot u$. Then we verify that all the divisors of $t^s \cdot u \pmod q$ belong to the factor bases; that means, we have already obtained the discrete logarithms of them. If it does not hold, we need to choose another s. Once s is found, we get

$$t^s \cdot u = \textstyle\prod_{i=1}^r c_i^{d_i} \pmod q.$$

By applying the logarithm on both sides we obtain

$$\log_t(t^s \cdot u) = \log_t(t^s \cdot t^x) \equiv \textstyle\sum_{i=1}^r d_i \log_t(c_i) \pmod q,$$

which yields

$$s + x \equiv \textstyle\sum_{i=1}^r d_i \log_t(c_i) \pmod q.$$

As we already know the values of the discrete logarithms $\log_t(c_i)$, we can now easily compute x, which is our searched discrete logarithm.

Another very important characteristic of the `NFS-DLP` algorithm needs to be mentioned. When we want to compute multiple discrete logarithms in the same field we can consider the choice of the polynomials and the sieving stage as pre-computation steps that we need to do just once. We can reuse the computed discrete logarithms of the elements from the factor bases for all searched discrete logarithms from the same field. This is the biggest difference because in the `NFS` for the factorization, we need to calculate all the steps again for every new number we want to factorize.

Conclusion

The goal of this work was to analyze the number field sieve algorithm (`NFS`) for discrete logarithm and present the differences between the `NFS` for discrete logarithm and the `NFS` for factorization.

We covered the important theory – number fields, Dedeking Domains and factorization of ideals. We presented the Index Calculus algorithm that is the base for both `NFS` algorithms and for the illustration we provided an example of the Index Calculus algorithm using integers. We investigated the `NFS` for discrete logarithm in more detail with focus on the sections that are different from the `NFS` for factorization. We also computed two examples of the `NFS` for discrete logarithm, one for $\mathbb{F}_p$ and one for $\mathbb{F}_{p^n}$. All our examples were calculated using computer algebra system PARI/GP and the scripts are attached to this work. As our aim was to compare the `NFS` for discrete logarithm and for factorization, we also described the number field sieve algorithm for factorization and finally, we analyzed the differences.

The number field sieve algorithms for factorization and for discrete logarithms seem similar as they are both based on the number fields and on the same principle coming from the Index Calculus algorithm. But when one starts investigating them in more detail, substantial differences are found.

In the discrete logarithm case, when computing in the number field of the class number one, the matrix composed of the relation also contains the units coming from the conversion of the ideals to elements. When this matrix is solved, the discrete logarithms of the elements from the factor bases are directly obtained. To compute the searched discrete logarithm, precomputed discrete logarithms are used.

In the factorization case, the matrix composed of the relation also contains the quadratic characters to guarantee the existence of the square roots. Once this matrix is solved, the square roots have to be computed to obtain the values needed in the final step. To find the divisor of a number, the greatest common divisor is computed from the difference of our obtained values and the given number.

One more difference is found when looking at the properties of the algorithms. When computing multiple discrete logarithms in the same field, the choice of polynomials and the sieving stage can be considered as a pre-computation. The calculated discrete logarithms of the elements of the factor bases can be reused. In the factorization, with every new number that we want to factorize, we need to run all the steps of the algorithm again.

To conclude, even though the `NFS` algorithms for discrete logarithm and for factorization seem very similar, some steps are different. More differences arise when one thinks about the implementation, which can be considered for the future work. Small details such as computing a matrix in $\mathbb{Z}_2$ or in $\mathbb{Z}$ make huge impact on the implementation.

Bibliography

[1] Daniel M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discret. Math.*, 6(1):124–138, February 1993.

[2] Leonard M. Adleman. The function field sieve. In Leonard M. Adleman and Ming-Deh Huang, editors, *Algorithmic Number Theory*, volume 877 of *Lecture Notes in Computer Science*, pages 108–121. Springer Berlin Heidelberg, 1994.

[3] Leonard M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, SFCS '79, pages 55–60, Washington, DC, USA, 1979. IEEE Computer Society.

[4] Andrew M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Proc. Of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*, pages 224–314, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[5] Daniel A. Marcus. *Number Fields.* Springer-Verlag, 1977.

[6] Robert B. Ash. *A Course in Algebraic Number Theory.* Dover Publications, Mineola, New York, 2010.

[7] Henri Cohen, editor. *A Course in Computational Algebraic Number Theory.* Graduate Texts in Mathematics. Springer-Verlag, Berlin, 1993.

[8] Raminder Ruprai. Improvements in the index-calculus algorithm for solving the discrete logarithm problem over F_p. `http://www.isg.rhul.ac.uk/~prai175/ISGStudentSem07/IndexCalculus.pdf`, October 2007.

[9] Urs Wagner. Wiedemann's algorithm. `http://www.math.uzh.ch/?file&key1=9252`, 2008.

[10] Antoine Joux and Reynald Lercier. Number field sieve for the DLP. In Henk C.A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 867–873. Springer US, 2011.

[11] Maurice Kraitchik. *Theorie des nombres*, volume 1. Gauthier-Villars, Paris, 1922.

[12] Maurice Kraitchik. *Recherches sur la theorie des nombres.* Gauthier-Villars, Paris, 1924.

[13] A. E. Western and J. C. P. Miller. *Tables of indices and primitive roots.* Royal Society Mathematical Tables. Cambridge University Press, Cambridge, 1968.

[14] Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford University, Stanford, CA, USA, 1979.

[15] Martin E. Hellman and Justin M. Reyneri. Fast Computation of Discrete Logarithms in GF(q). In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 3–13. Plenum, 1982.

[16] Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30(4):587–594, September 2006.

[17] Taher El Gamal. A Subexponential-Time Algorithm for Computing Discrete Logarithms over GF(p^2). In David Chaum, editor, *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 275–292. Plenum, New York, 1984.

[18] Don Coppersmith, Andrew M. Odlyzko, and Richard Schroeppel. Discrete Logarithms in GF(p). *Algorithmica*, 1(1):1–15, January 1986.

[19] A. K. Lenstra and H. W. Lenstra, Jr., editors. *The Development of the Number Field Sieve.* Lecture notes in mathematics. Springer, New York, 1993.

[20] Leonard M. Adleman and Jonathan Demarrais. A subexponential algorithm for discrete logarithms over all finite fields. *Mathematics of Computation*, 61(203):1–15, 1993.

[21] Oliver Schirokauer. Discrete logarithms and local units. *Philosophical Transactions: Physical Sciences and Engineering*, 345(1676):409–423, 1993.

[22] Oliver Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comput*, 69(231):1267–1283, 2000.

[23] Don Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6(3):169–180, 1993.

[24] Leonard M. Adleman and Ming-Deh A. Huang. Function field sieve method for discrete logarithms over finite fields. *Inf. Comput.*, 151(1-2):5–16, May 1999.

[25] Antoine Joux and Reynald Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. *Mathematics of Computation*, 72(242):953–967, 2003.

[26] Oliver Schirokauer. The impact of the number field sieve on the discrete logarithm problem in finite fields. *Algorithmic Number Theory. Mathematical Sciences Research Institute Publications*, 44:397–420, 2008.

[27] Antoine Joux, Reynald Lercier, Nigel Smart, and Frederik Vercauteren. The number field sieve in the medium prime case. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, CRYPTO'06, pages 326–344, Berlin, Heidelberg, 2006. Springer-Verlag.

[28] J.-M. Couveignes. *The Development of the NFS*, chapter Computing a Square Root for the Number Field Sieve, pages 95–102. Springer-Verlag, New York, 1993.

[29] P. L. Montgomery. Square Roots of Products of Algebraic Numbers. In W. Gautschi, editor, *Mathematics of Computation 1943-1993: a Half-Century of Compurational Mathematics*, Proc. of Symposia of Applied Mathematics, pages 567–571. American Mathematical Society, 1994.

[30] Vojtěch Matocha. Pokročilé metody hledání diskrétního logaritmu. Master's thesis, Charles University in Prague, 2013.

[31] Lukáš Perůtka. Hledání optimálních strategií číselného síta. Master's thesis, Charles University in Prague, 2009.

Attachments

CD Content

The enclosed CD contains scripts used to compute the Index Calculus algorithm and the `NFS` for discrete logarithm:

- findPolClassNumber-Zp.gp - The script finds the appropriate polynomials that create the number fields of the class number one and can be used in the NFS for $\mathbb{F}_p^*$.
- findPolClassNumber-Zp^n.gp - The script finds the appropriate polynomials where at least one polynomial creates the number fields of the class number 1 and can be used in the NFS for $\mathbb{F}_{p^n}^*$.
- findPrimeIdeals.gp - The program finds the prime ideals of two number fields given by polynomials.
- functions.gp - This file contains all functions needed to compute the NFS for discrete logarithm.
- index_calculus.gp - This program computes the discrete logarithm x, ($t^x = u \bmod p$), in finite cyclic group $\mathbb{Z}/p\mathbb{Z}$, where p is a prime number, u is a given number and t is a primitive element of the group.
- nfs_dlp.gp - This algorithm computes random coprime pairs (a, b), factorizes them and composes the bases of K_1 and K_2 from the factorization of these pairs and creates the matrix from these relations.
- nfs_dlp2.gp - This algorithm computes coprime pairs (a, b), such that its factors are in given base1 and base2. It will also compose the matrix of the relations.
- testFunctions.gp - This file contains couple of tests of functions from functions.gp file.
- testing_data_index_calculus - Contains testing data with results for Index Calculus script.

Index Calculus Algorithm in PARI/GP: Two Main Functions

This section shows two main functions of the Index Calculus algorithm implemented in PARI/GP to compute the discrete logarithm x such that $t^x = u \pmod p$ in finite cyclic group $\mathbb{Z}/p\mathbb{Z}$, where p is a prime number, u is a given number and t is a primitive element of the group. We denote by S the size of a factor base.

```
\\ computes discrete logarithm of the elements of a factor base
ComputeSmallLogarithms(t,p,S)=
{
  my(actNumRel = 0, i = random(p-2), k = 0, T, mat, solVec, ind,
     discLogs);

  \\ algorithms will compute 3*|S| relations. Some algorithms
     present constant 2.
  numRel = round(3 * length(S));
  solVec = vector(numRel);
  mat = vector(length(S));
  solVec = vector(numRel,index,0);

  \\ while we do not have enough relations
  while(actNumRel < numRel,

    \\set up the relation vector to 0
    relation = vector(length(S),index,0);

    \\ check if t^i is B-smooth
    for(j = 1,matsize(T = Func1(i,t,1,p))[1],
      if(ind = setsearch(S,T[j,1]),  \\ ind is index of prime in
         S
        relation[ind] = T[j,2], \\ setup the value in relation
           vector
        k=1)); \\end of for

    \\if all factors are in S, so a^i is B-smooth, add relation
       vector to the matrix and update the solution vector
    if(k == 0 && actNumRel > 0, mat = matconcat([mat;relation]);
       actNumRel = actNumRel + 1; solVec[actNumRel] = i);
    if(k == 0 && actNumRel == 0, mat = relation; actNumRel =
       actNumRel + 1; solVec[actNumRel] = i);

    i = random(p-2);
    k = 0);\\end of while

    \\check if det mat~*mat is == 0. If yes, repeat whole process
        of putting together the matrix and solution vector again
\\    if(actNumRel == numRel,
\\      if(matdet((mat~)*mat)==0,
\\        actNumRel = 0;
\\        solVec = vector(numRel,index,0);
\\        relation = vector(length(S),index,0);
\\        mat = vector(length(S), index, 0))));\\end of while

  \\ solve the matrix
  SolveMatrix(mat, solVec, p);
}

\\ computes the searched discrete logarithm x from precomputed
    discrete logarithms of the elements from the factor base
ComputeLogarithm(t,u,p,S,smallLogs)=
{
  my(k = 1, i, output=0, A);

  while(k != 0,
    k = 0;
    i = random(p-2);

    \\ Find the exponent of t, such that u*t^exponent mod p
       factors to the prime numbers that are in factor base S.
       Denote the exponent by i. Then it holds that log_t(u) = c1
       *log_t(p1) + .. + cn*log_t(pn) - i.
    \\ A contains the factorization of u*(t^i) mod p. First
       column contain factors, second column contains
       multiplicity of this factor
    for(j = 1,matsize(A = Func1(i,t,u,p))[1],
      if(setsearch(S,A[j,1]),
        \\print(i,"+",j,",prime=",A[j,1]),
        ,
        k = 1));
    if(k == 0, print("Desired discrete logarithm will be
       calculated from these factors: ", A))); \\end of while

  \\ compute the discrete logarithm using factorization of the
     previous step by computing discrete logarithms of the factor
      elements
  for(l = 1,matsize(A)[1],output = output + (A[l,2]*smallLogs[ind
      = setsearch(S,A[l,1])] % (p-1)));

  output = ((output % (p-1)) - i % (p-1));
  if(output < 0, output = output+(p-1), output = output);
}
```

NFS for Discrete Logarithm in PARI/GP: Finding Relations

This script is computing pairs (a, b) such that the ideal $(a + b\theta_i)$ factorizes to the elements of the given bases of K_1 and K_2.

```
Run()=
{
  \\ import functions from functions.gp file
  read("functions.gp");

  p = 173;
  n = 1;

  f1 = x^2 + 83;
  f2 = x^2 + 83 + p;

  \\smoothness bounds for both factor bases
  smoothBoundK1 = 30;
  smoothBoundK2 = 30;

  \\ sieve limit
  S1 = 200; \\ |a| <= S1
  S2 = 200; \\ 0<= b <= S2

  \\ create number fields
  K1=bnfinit(f1);
  K2=bnfinit(f2);

  base1 = listcreate;
  base2 = listcreate;

  \\ given bases
  base1 = [[3, [0, 2]~], [3, [2, 2]~], [7, [0, 2]~], [7, [2,
     2]~], [11, [-3, 2]~], [11, [5, 2]~], [17, [-5, 2]~], [17,
     [7, 2]~], [23, [-2, 2]~], [23, [4, 2]~], [29, [-1, 2]~],
     [29, [3, 2]~]];
  base2 = [[2, [1, -1]~], [5, [-2, 16]~], [5, [2, 16]~], [13,
     [-2, 16]~], [13, [2, 16]~], [17, [-4, 16]~], [17, [4, 16]~],
      [29, [-11, 16]~], [29, [11, 16]~]];

  pairs = listcreate;
  pairsFactors = listcreate;
  actNumberOfPairs = 0;
  numberOfPairs = 22;

  numberOfPairsTried = 0;

  while(actNumberOfPairs < numberOfPairs,
    \\ choose random a and b s.t. |a| <= S1 and 0<= b <= S2
    a = random(2*S1 + 1) - S1; \\print("a: ",a);
    b = random(S2 + 1); \\print("b: ",b);
    numberOfPairsTried = numberOfPairsTried + 1;

    \\ check, if [a,b] is not coprime and is already in pairs,
       and if yes, choose another one while it is coprime and is
```

```
        not already in pairs
    \\ [1,0] pair gives zero row in matrix - we want to avoid
        that
    while(!(setsearch(pairs,[a,b]) == 0 && gcd(a,b) == 1 && (a !=
         1 && b != 0) && (a != -1 && b != 0) && IdealIsBsmooth(K1,
         a + b*x, smoothBoundK1) && IdealIsBsmooth(K2, a + b*x,
        smoothBoundK2)),
      a = random(2*S1 + 1) - S1;
      b = random(S2 + 1);
      numberOfPairsTried = numberOfPairsTried + 1); \\ end of
         while

    k = 0;
    \\ check smoothness of the ideals
    \\ FactorsAreInBasis function is from functions.gp file
    if(FactorsAreInBasis(fact1=idealfactor(K1,a + b*x), base1) &&
         FactorsAreInBasis(fact2=idealfactor(K2,a + b*x), base2),
      \\ create list of pairs (a,b) and their factorization in
         the number fields
      listput(pairs,[a,b]);
      listsort(pairs);
      listput(pairsFactors,[a,b,fact1, fact2]);
      listsort(pairsFactors);

      actNumberOfPairs = actNumberOfPairs +1); \\ end of if
  ); \\end of while

  columnsNum = length(base1) + length(base2);
  \\ CreateMatrix function is from functions.gp file
  mat = CreateMatrix(columnsNum, pairsFactors, base1, base2);

  \\ WriteGenerators function is from functions.gp file
  print("generators in base1: ");
  print(WriteGenerators(K1,base1));
  if(K1.clgp.no == 1,
    print("Compute multiplication of generators for all elements
        for K1: ");
    print(ComputeMulOfGenList(K1, base1, pairs)));

  print("generators in base2: ");
  print(WriteGenerators(K2,base2));
  if(K2.clgp.no == 1,
    print("Compute multiplication of generators for all elements
        of K2: ");
    print(ComputeMulOfGenList(K2, base2, pairs)));
  print("base1: ", base1);
  print("base2: ", base2);
  print("List of pairs: ", pairs);

}
```