Improvements on the Discrete Logarithm Problem in GF(p)

Razvan Barbulescu

École Normale Supérieure de Lyon

Mémoire de stage de recherche en deuxième année de master

Improvements to the Discrete Logarithm Problem in $\mathbf{F}_p^*$

Auteur:
Răzvan Bărbulescu

Directeur:
Pierrick Gaudry

4 mars 2011

Laboratoire Lorrain de Recherche en Informatique et ses Applications

Contents

Chapter 1

Introduction

For long, number theorists have been puzzled by the equation $T^x \equiv S \bmod p$. One calls x the index or the discrete logarithm of S in base T, and computing the index is called discrete logarithm problem, or DLP in short. Bouniakowsky [Bou70] gave the first algorithmic solution in 1870 and half a century later Kraitchik [Kra22] created the first sub-exponential method. Nevertheless, Kraitchik's work was forgotten and in 1971 Shanks gave a new solution called Baby-Step-Giant-Step with poorer performance: $\mathcal{O}(p^{\frac{1}{2}})$. In the 70s everything made computer scientists believe that the DLP was a difficult problem[1]. Thus in 1976 Diffie and Hellman used exponentiation in $(\mathbb{Z}/p\mathbb{Z})^*$ as a potential one-way function and thanks to this example showed that the public key cryptography is feasible. Later came DSS and ECC which rely on the difficulty of DLP in $\mathbb{F}_{p^n}^*$ and on the elliptic curves, respectively.

In what follows we consider exclusively the case of $\mathbb{F}_p^*$ since a couple of issues make it the most important case for DLP. First, Bach [Bac84] proved that the DLP in $(\mathbb{Z}/N\mathbb{Z})^*$ is equivalent, by polynomial-time reductions, to the pair of problems (factorize N and solve discrete logarithm for all prime factors p of N); moreover the Silver-Pohlig-Hellman algorithm [PH78] makes reduction from $(\mathbb{Z}/N\mathbb{Z})^*$ to $\mathbb{F}_p^*$ effective. Next, every time a fast algorithm was designed for $\mathbb{F}_p^*$, it was adapted to the case $(\mathbb{F}_{p^n})^*$ with no loss of speed. Even though the converse is not true since Coppersmith designed an algorithm for $(\mathbb{F}_{2^n})^*$ faster than those for $\mathbb{F}_p^*$, one continues to use the case of prime fields as a starting point for the case of the most difficult finite fields, i.e. when $\log(p)$ is large compared to n. Finally, in the case of the elliptic curves no sub-exponential general purpose attack was published in the 25 years since ECC was created, but all the attempts try to replicate Index Calculus.

[1] Diffie and Hellman write in [DH76] "However we assume that the best known algorithm for computing logs mod q [i.e. Giant-Step-Baby-Step] is in fact close to optimal and hence $q^{\frac{1}{2}}$ is a good measure of the problem's complexity, for properly chosen q".

Contrary to the case of factorization in which the input is a composite number N, in the case of DLP we are given a prime number p, a generator T of $\mathbb{F}_p^*$ and an element S of $\mathbb{F}_p^*$. Therefore the DLP based cryptosystems fix p and T and leave to S the role of secret piece of information. But in 2003 Joux and Lercier [JL03] implemented an algorithm which significantly speeds up the computations if p is known in advance. The algorithms which followed have two parts: one which depends only on p called precomputations and one which depends on T and S called individual logarithm. Notice that this structure of algorithms is also useful in a number theoretical context where one might want to compute the discrete logarithm for a unique p and a set of values of S. In 2006 Commeine and Semaev [CS06] published an algorithm with complexity $L_p(\frac{1}{3}, 1.902)$ for the precomputations and $L_p(\frac{1}{3}, 1.442)$ for the individual logarithm. Remember the traditional notation:

$$L_x(s, c) = \exp(c(\log x)^s(\log\log x)^{1-s})$$

where log denotes the natural logarithm. Sometimes we drop x and/or the second constant and write $L(s)$ or $L_x(s)$. When speaking about algorithmic complexity we write $L_x(s, c)$ instead of $L_x(s, c)^{1+o(1)}$.

In this paper we improve Commeine and Semaev's algorithm in two ways. First we show a method to speed up the precomputations to $L_p(\frac{1}{3}, 1.639)$ using an idea of Coppersmith [Cop93] called Factorization Factory. For now, this algorithm is of theoretical interest only because, for computing DLP for values of p in the range $p_0 < p < p_0 \cdot L_{p_0}(\frac{2}{3}, 0.905)$, it requires a permanent storage space of size $L_{p_0}(\frac{1}{3}, 1.639)$ computed in time $L_{p_0}(\frac{1}{3}, 2.007)$. Later we present a way to speed up the individual logarithm to $L_p(\frac{1}{3}, 1.232)$. This idea works as well in theory as in practice and is inspired from a paper of Pomerance [Pom82].

The second chapter is a short overview of many algorithms related to factorization and DLP. This can be used both as a general introduction to the field or to the state-of-art algorithms for DLP in particular. See the graph at the end of the report for the relations between different algorithms. In the third chapter we give full details for the individual logarithm. In particular we explain the early abort strategy and prove two theorems of analytic number theory.

Chapter 2

State of Art

2.1 Index Calculus

The main idea of all the sub-exponential algorithms for discrete logarithm is as follows: start by collecting linear relations between discrete logarithms of several elements, next solve the linear system obtained and finally express the discrete logarithm of S in terms of the computed discrete logs. Discovered for the first time in 1922 by Kraitchik [Kra22], the idea was rediscovered in connection with the Diffie-Hellman cryptosystem in 1977 [Poh77],[Adl08]. A key notion for the algorithm is smoothness.

Definition 1 *Let $\mathcal{P}$ be a set of prime numbers that we call factor base. An integer n is $\mathcal{P}$-smooth if all the prime factors of n are in $\mathcal{P}$. If $\mathcal{P}$ is the set of prime numbers less than a bound B then we say n is B-smooth.*

The shape of the algorithm below deserves particular attention because many recent algorithms speeded up each step of Index Calculus rather than changing the big lines. We call p_i the i^{th} prime number.

Algorithm 1 *Index Calculus*

$INPUT$: $T, S \in \mathbb{F}_p^*$ such that T spans $\mathbb{F}_p^*$; a prime factor q of $p-1$.

$OUTPUT$: $log_T S$ modulo q.

0. *Set $B \leftarrow [L_p(\frac{1}{2}, \frac{1}{\sqrt{2}})]$, $R \leftarrow \pi(B)$;*
1. *Choose and check random h until R values of h are found such that Remainder(T^h,p) is B-smooth; write each equation of type $T^h \ mod\ p = p_1^{e_1} \dots p_R^{e_R}$ in additive form:*

$$h = e_1 \log_T(p_1) + \ldots + e_R \log_T(p_R) \ mod\ q; \tag{2.1}$$

2. *Solve the sparse linear system of size $R \times R$ with solution $\{log_T l \ mod \ q |$ where $l \leq B$ is prime $\}$;*

3. *Choose and check random h until one value is found such that Remainder($T^h S$,p) is B-smooth; this gives a relation of the form:*

$$T^h S \ mod \ p = p_1^{f_1} \dots p_R^{f_R}; \tag{2.2}$$

4. *Output $x = (f_1 \log_T p_1 + \dots + f_R \log_T p_R - h) \ mod \ q$.*

Notice that the input and the output are not the most general possible. Indeed, one can ask for $\log_T S \bmod (p-1)$ with $S \in \langle T \rangle \neq \mathbb{F}_p^*$. It is solved by doing as follows: factor $p-1$; find a generator ρ of $\mathbb{F}_p^*$; compute $\log_\rho T$ and $\log_\rho S$ for each prime factor q of $p-1$; compute by the Chinese Remainder Theorem an l such that $l \equiv \frac{\log_\rho S}{\log_\rho T} \bmod q$ for all q; output l. One finds ρ by random picks since the are $\varphi(p-1)$ generators in $\mathbb{F}_p^*$ and a raw inequality is $\frac{\varphi(p-1)}{p-1} > \frac{1}{\log(p-1)+5}$. In practice we test the values $2, 3, \dots$ in a row, which is justified by Shoup [Sho90] if the Extended Riemann Hypothesis is true. It also allows us to consider T smooth in the sequel.

2.2 Smoothness

One can expect the smooth numbers to be rare. Still, they are many enough to be used in DLP algorithms and we can estimate their number. Let's denote $\psi(x, B)$ the number of integers smaller than x which are B-smooth. The following theorem of Candfield, Erdös and Pomerance published in 1983 estimates $\psi(x, B)$.

Theorem 1 *Let ϵ be a positive constant. Then it holds:*

$$\psi(x, B)/x = u^{-u(1+o(1))}$$

uniformly in the region $x \geq 10$ and $B \geq (\log x)^{1+\epsilon}$, where $u = \frac{\log x}{\log B}$ and the limit implicit in the $o(1)$ is for $u \to \infty$.

proof as theorem 3.1 in [CEP83].

Corollary 1 *Let $x = L_p(a, b)^{1+o(1)}$ and $B = L_p(c, d)^{1+o(1)}$ with $(c, d) < (a, b)$ in lexico-graphical order. Then $\frac{\psi(x,B)}{x} = L_p(a-c, (a-c)\frac{b}{d})^{-1+o(1)}$.*

Smoothness would not be such an interesting property without an efficient way to test it. The best algorithm for this purpose is the Elliptic Curves Method, abbreviated ECM. Indeed, let's call x the number to test and B the smoothness bound. Let's call an ECM run to launch ECM on as many curves as we can in

time $L_B(\frac{1}{2}, \sqrt{2}) \cdot (\log x)^{\mathcal{O}(1)}$. As explained by Lenstra in [Len87], the result can be either a proper factor of the number or "not smooth" which is correct with probability 1/2. If we find a factor we make a next run and continue until we find the answer "not smooth" or we find all the factors of x. Since the total number of factors of x is less than $\log_2 x$, it doesn't count in L-notation. Therefore the total time of a smoothness test is $L_B(\frac{1}{2}, \sqrt{2}) \cdot (\log x)^{\mathcal{O}(1)}$. In the case of NFS we have $B = L_p(c, d)^{1+o(1)}$ and the smoothness test takes time $L_p(\frac{c}{2}, \sqrt{2cd})^{1+o(1)}$.

2.3 Sparse Systems

Since one deals with large linear systems, even a minor speedup of the algorithms for solving linear systems has a significant asymptotic effect. The naive method, i.e. Gaussian elimination, has complexity $\mathcal{O}(n^3)$ while the best algorithm for solving linear systems has a complexity of $\mathcal{O}(n^w)$ with $w \approx 2.49$ (see [CW08]). Still, the system in Index Calculus is sparse, i.e. it cannot have more than $\mathcal{O}(\log(p))$ nonzero entries per line. Indeed, the lines of the matrix in the linear system are the valuation vectors of numbers of size p in a factor base. Since each prime is larger than 2, a number of size p has at most $\log_2(p)$ prime divisors by counting multiplicities. Therefore, each line of the linear system has at most $\mathcal{O}(\log(p))$ nonzero entries. Several algorithms like [Wie86] were designed to solve sparse systems in $\mathcal{O}(n^2 w)$ where w is the the weight, i.e. the number of nonzero entries per line. In the first approximation, for the algorithms evolved from Index Calculus, if the system solved has size $L_p(s, c) \times L_p(s, c)$ then the linear algebra costs $L_p(s, c)^{(1+o(1))\cdot 2} \cdot \log p = L_p(s, 2c)^{1+o(1)}$.

2.4 Complexity Analysis

Pomerance writes in [Pom08] that he created the Quadratic Sieve while looking at the complexity analysis of former algorithms and thinking about which step can go faster. In order to compute the complexity of Index Calculus we consider the following problem:

Problem 1 *Let $N \in \mathbb{N}$ be given. Assume one can use a generator of random numbers of size $L_N(\sigma, \gamma)$ with $\sigma \in (0, 1)$ and $\gamma > 0$ fixed. Assume also that one can inspect for smoothness in negligible time. Choose convenient parameters $s \in (0, 1)$ and $c > 0$ and design an algorithm which in optimal time finds $[L_N(s, c)]$ numbers $L_N(s, c)$-smooth and then solves a sparse linear system of size $L_N(s, c)$.*

Index Calculus spends part of its time on solving problem 1 for $N = p$, $\sigma = 1$ and $\gamma = 1$ by a choose&check strategy. Indeed, step 1 solves the first part of

the problem because we find $\pi(L_p(s,c))$ relations with $\pi(x) = \#\{p\ prime \leq x\}$ and because according to the Prime Number Theorem $\pi(x) \sim \frac{x}{\log x}$ which is x in L notation. Next, step 2 solves a sparse linear system of size $\pi(L_p(s,c)) = L_p(s,c)^{1+o(1)}$.

The time needed for step 1 is $P_{smooth}(L_p(\sigma,\gamma), L_p(c,s))^{-1} \cdot L_p(s,c) \cdot t_{test}$ with t_{test} the time of a smoothness test. At the time Index Calculus was published t_{test} used to be important because the test was trial division. But thanks to ECM testing smoothness became sub-exponential, so we can neglect it in L notation. According to the section Sparse Systems step 2 takes time $L_p(s,c)^2 = L_p(s,2c)$. Therefore steps 1 and step 2 in Index Calculus have complexity:

$$L_p(\sigma - s, (\sigma - s)\frac{\gamma}{c})L_p(s,c) + L_p(s,2c). \tag{2.3}$$

We minimize this expression for fixed σ and γ by taking $s = \frac{\sigma}{2}$ and $c = \sqrt{\frac{\sigma\gamma}{2}}$. Since $\sigma = 1$ and $\gamma = 1$, time(step 1 + step 2)$=L_p(\frac{1}{2}, \sqrt{2})$. One checks that step 3 is negligible, so

$$time(Index\ Calculus) = L_p(\frac{1}{2}, \sqrt{2})^{1+o(1)}.$$

Remark 1 *Every time an algorithm was designed for factorization using smoothness and linear algebra, there was an algorithm for DLP with the same complexity. Nevertheless it does not imply the two problems are connected. It is rather a consequence of reducing both problems to instances of same size of problem* 1.

2.5 Sieving

We decide to replace choosing random numbers and testing their smoothness by a routine which finds smooth numbers all at once. In this way we save the time for smoothness testing even though it does not count in the L notation of the complexity (and unfortunately uses more space)[1]. The procedure goes as it follows. First make an array in which the n^{th} position records $[\log_2(n)]$. Then for each prime number p (first untouched entry of the array, not p in DLP) subtract $[\log_2 p]$ from $p, 2p \ldots$, then subtract $\log_2 p$ form $p^2, 2p^2, \ldots$ and so on. At the end, the entries of the array which are almost zero correspond[2] to smooth numbers. Let's call A the cardinal of the sieving array and we impose the condition $A \geq B$. Then the time needed for a sieve is $A \cdot (\sum_{\substack{p\ \text{prime} \\ p<B}} \frac{1}{p})$ which leads[3] to a

[1] Pomerance used this idea from Schroeppel to create the Quadratic Sieve. It was a significant advance because at that time smoothness used to be tested by trial division.

[2] See [Sil87] for a precise practical analysis.

[3] Use the formula $\sum_{\substack{p\ \text{prime} \\ p<x}} \frac{1}{p} \sim \log\log x$, an easy corollary of the Prime Number Theorem.

complexity $A \cdot \log\log(B)$ which is $A^{1+o(1)}$. If we want to find $[L_p(s,c)]$ numbers of size $L_p(\sigma,\gamma)$ which are $L_p(s,c)$-smooth then we need to satisfy the condition $P_{smooth}(L_p(\sigma,\gamma), L_p(s,c))A \geq L_p(s,c)$. Thus the complexity for solving problem 1 with the sieve is $A^{1+o(1)} = L_p(s,c) \cdot P_{smooth}(L_p(\sigma,\gamma), L_p(s,c))^{-1}$ which is exactly the complexity of the choose&check strategy.

Next, if we want smooth values for a polynomial $f \in \mathbb{Z}[X]$ we can sieve on pairs (x,y) such that $|x|, |y| \leq \sqrt{A}$. For each x_0 such that $|x_0| \leq \sqrt{A}$ and each prime p we find in polynomial time of $deg(f)$ and p an $y_0 \leq p$ such that $p \mid f(x_0, y_0)$. Then we sieve on $(x_0, y_0), (x_0, y_0 + p), (x_0, y_0 + 2p) \ldots$. One can read a detailed proof in [Sch93].

2.6 Double smoothness

An important idea is to replace a relation of type $t^h \bmod p = p_1^{e_1} \ldots p_r^{e_r}$ by two half-equations as below[4]:

$$\begin{cases} \gamma = p_0^{e_0} p_1^{e_1} \ldots p_r^{e_r} \\ \gamma = q_0^{f_0} q_1^{f_1} \ldots q_s^{f_s} \end{cases} \tag{2.4}$$

The advantage is when γ, the number we inspect for smoothness, is significantly smaller so we need to repeat the choose&check step less times. In order to obtain two equations like (2.4) one can use for instance imaginary quadratic fields[5], i.e $\mathbb{Q}[\sqrt{-s}]$ with $s \in \mathbb{N}$. Suppose one wants to compute the discrete logarithm in $\mathbb{F}_p$. One picks r in $\{-1, -2, -3, -7, -11, -19, -43, -67, -163\}$[6] such that $X^2 - r \equiv 0 \bmod p$ has roots in $\mathbb{F}_p$. For example there is 50% chance for one to have $r = -1$ in which case one deals with $\mathbb{Z}[i]$, the Gaussian integers. If no value of r works we use a different algorithm.

The factor base of prime numbers we had in Index Calculus is replaced here by two factor bases: one of prime numbers less than B and one of prime elements in $\mathbb{Z}[\sqrt{r}]$ with norm less than B with $B = [L_p(\frac{1}{2}, \frac{1}{2})]$. Together, the two factor bases have less than B elements since for each rational prime l there are at most

[4]The Gaussian Integers Algorithm is described briefly in [COS86]. It is noticeable that in the same article the authors describe in detail another algorithm of same complexity in the first approximation, but a worse one in a closer look. The better complexity for the Gaussian Integers Algorithm was confirmed in practice since it was the fastest DLP algorithm between 1986 and 1998.

[5]The quadratic fields have been used for the first time to compute discrete logarithm in $GF(p^2)$. See [ElG02]

[6]The list is such that $\mathbb{Z}[\sqrt{r}]$ is a unique factorization domain according to Class Number Problem of Gauss which was solved by Stark [Sta07].

two primes of $\mathbb{Z}[\sqrt{r}]$ with norm divisible by l and because $\pi(B) < \frac{B}{3}$ for $B > 10$. The pleasant property of $\mathbb{Z}[\sqrt{r}]$ is that the norm is multiplicative, i.e. by putting $N(a+b\sqrt{r}) = a^2 - rb^2$ we have $N(xy) = N(x)N(y)$ for all $x, y \in \mathbb{Z}[\sqrt{r}]$. This means we test for smoothness by simply testing the norm for smoothness. Moreover, if we want to factorize an element x we start by factorizing $N(x)$ and then we compute for each prime $l \mid N(x)$ the irreducible elements $l_i \in \mathbb{Z}[\sqrt{r}]$ such that $l_i \mid a$. Indeed, the condition $l_i \mid a$ is equivalent to a system of two linear equations.

The algorithm starts by computing a generator $b = e + f\sqrt{r}$ of the unit group of $\mathbb{Z}[\sqrt{r}]$ which is added to the factor base. Next it finds a root X_0 of $X^2 - r \equiv 0 \bmod p$, computes a pair $(T, V) \in \mathbb{Z}^2$ such that $T \equiv X_0 V \bmod p$ and $0 \leq T, V < \sqrt{p}$[7] and adds V to the factor base. Let us call $E = L_p(\frac{1}{2}, \epsilon)$ for some parameter ϵ. The algorithm continues by sieving for values of $(c, d) \in [-E, E]$ such that both $cT_0 + dV_0$ and $\mathrm{N}(c + \sqrt{r}d)$ are B-smooth. Since $\mathbb{Z}[\sqrt{r}]$ is here a unique factorization domain and b is a generator of the unit group one can write

$$c + d\sqrt{r} = b^{f_0} l_1^{f_1} \dots l_s^{f_s} \tag{2.5}$$

for l_i irreducible(prime) elements in $\mathbb{Z}[\sqrt{r}]$. Since $\mathrm{N}(c + \sqrt{r}d) = \mathrm{N}(l_1)^{f_1} \dots \mathrm{N}(l_s)^{f_s}$ and $\mathrm{N}(c + \sqrt{r}d)$ is B-smooth, the l_i are in the factor base. Similarly, since $cT_0 + dV_0$ is B-smooth, one can write:

$$cV + dT = (-1)^{e_0} p_1^{e_1} \dots p_k^{e_k} \tag{2.6}$$

with the p_i in the rational factor base. Since $T = X_0 V$, we have $cV + dT = V(c + X_0 d)$. Put $\phi : \mathbb{Z}[\sqrt{r}] \to \mathbb{Z}/p\mathbb{Z}$, $a + b\sqrt{r} \mapsto a + bX_0$. It is a ring homomorphism because both $X_0 \in \mathbb{F}_p$ and $\sqrt{r} \in \mathbb{C}$ are roots of $X^2 - r$. The next system is (2.6) and $V \cdot \phi((2.5))$ and concerns elements of $\mathbb{Z}/p\mathbb{Z}$:

$$\begin{cases} V(c + dX_0) \equiv (-1)^{e_0} p_1^{e_1} \dots p_r^{e_r} \\ V(c + dX_0) \equiv V\phi(b)^{f_0} \phi(l_1)^{f_1} \dots \phi(l_s)^{f_s} \end{cases}$$

The linear algebra step finds the discrete logarithms of the p_i, the $\phi(l_i)$, $\phi(b)$ and V. This is possible because we don't start the linear algebra step until the number of equations exceeds that of unknown.

Where does the advantage come from? Since $cT_0 + dV_0 \leq 2\sqrt{p} \cdot E = L_p(1, \frac{1}{2})^{1+o(1)}$ and $\mathrm{N}(c + d\sqrt{r}) = \mathcal{O}(1)(c^2 + d^2) = L_p(1, \frac{1}{2})^{1+o(1)}$, the pair (c, d) is twice B-smooth with the probability that a number of size $\sqrt{p}$ is B-smooth. Thus the complexity of the Gaussian Integers algorithm is given by the equation 2.3 with $\sigma = 1$ and $\gamma = \frac{1}{2}$. Thus we have:

$$\text{complexity(Gaussian Integers)} = L(\frac{1}{2}, 1)^{1+o(1)}.$$

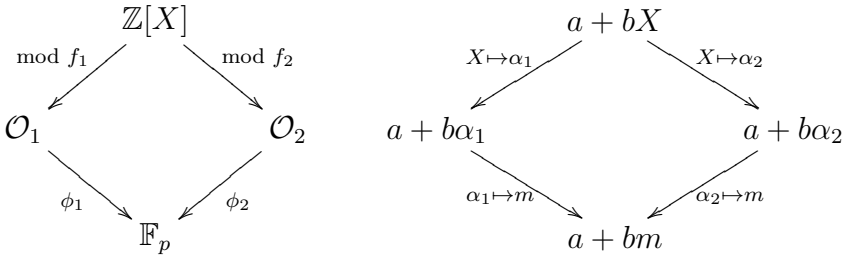
[7] Finding such T and V is called rational reconstruction of X_0 and takes polynomial time. For a proof use theorem 4.3 at page 58 in [Sho09].

2.7 Number Fields

In section Double smoothness we used fields K such that $[K : \mathbb{Q}] = 2$. Why not to use fields of higher degree? That's exactly what Pollard did in 1988 to factor $2^{128} + 1$, Fermat's 7^{th} number. It is a particular[8] case of NFS which we explain in the next lines.

Let $f_1, f_2 \in \mathbb{Z}[X]$ be two monic irreducible polynomials which share a root m in $\mathbb{F}_p$. In what follows $i = 1, 2$. Let's denote K_i the rupture field of f_i and let's call α_i a root of f_i in K_i. Let's call $\mathcal{O}_i$ the ring of integers in K_i. Put $\mathfrak{p}_i = (\alpha_i - m)\mathcal{O}_i + p\mathcal{O}_i$ which is obviously an ideal above p, i.e. which contains $p\mathcal{O}_i$. Theorem 2 below shows that $\mathfrak{p}_i$ is prime[9] and $\mathcal{O}_i/\mathfrak{p}_i \simeq \mathbb{Z}/p\mathbb{Z}$. Therefore one can define $\phi_i : \mathcal{O}_i \to \mathbb{Z}/p\mathbb{Z}$, $x \mapsto x \bmod \mathfrak{p}_i$. In particular for all $b_0, \ldots, b_{d-1}$ in $\mathbb{Z}$ we have $\phi_i(b_0 + b_1\alpha_i + \ldots + b_{d-1}\alpha_i^{d-1}) = b_0 + b_1 m + \ldots + b_{d-1}m^{d-1} \bmod p$.

Let $a + bX$ be in $\mathbb{Z}[X]$. Then $a + bX$ modulo f_i is in $\mathbb{Z}[X]/\langle f_i \rangle = \mathbb{Z}[\alpha_i] \subset \mathcal{O}_i$. Since $\phi_1(a + \alpha_1 b) = a + mb = \phi_2(a + \alpha_2 b)$, the diagram below is commutative.



How do we collect pairs of half-equations as in Double smoothness? Since in general $\mathbb{Z}[\alpha_i]$ is not an UFD, there is no canonical factorization into irreducible elements. Therefore we replace $\mathbb{Z}[\alpha_i]$ by $\mathcal{O}_i$ and prime elements by the prime ideals. In this way we keep the unique factorization property because $\mathcal{O}_{K_i}$ is a Dedekind domain. A sieve will find many couples (a, b) such that both $N_{K_1}(a+\alpha_1 b)$ and $N_{K_2}(a+\alpha_2 b)$ are smooth. This is possible because $N_{K_i}(a+\alpha_i b) = b^{deg(f_i)} f_i(-\frac{a}{b})$ is a polynomial in a and b.

But the unique factorization is not useful without a way to factorize a principal ideal $\delta\mathcal{O}$ into prime ideals. The good news is that there is an algorithm of Buchmann and Lenstra (6.2.2 in [Coh93]) which does so with the same[10] complexity we need to factor $N_K(\delta)$ into prime numbers.

[8] Pollard used $\mathbb{Z}[\sqrt[3]{2}]$ while in NFS we use arbitrary number fields.

[9] We need $p \nmid disc(f)$ which is easy to obtain.

[10] If factorization takes time T which is sub-exponential, Buchmann-Lenstra takes time $T \times (polynomial) = T^{1+o(1)}$ which is the same in L notation.

We can even forget about the Buchmann-Lenstra algorithm without changing the complexity in L notation. Indeed, as it is shown on the diagram 2.7 we only factor into ideals elements of the form $(a + b\alpha)$. We call badly ramified(or bad) for f a prime number l such that $l \mid [\mathcal{O}_K : \mathbb{Z}[\alpha]]$. The theorem below shows that if $N(a + b\alpha)$ does not have badly ramified factors then the time to factor $a + b\alpha$ is the time to factor $N(a + b\alpha)$ times an inversion *modulo p*.

Theorem 2 *Let a, b be coprime integers such that $N(a + \alpha b)$ has no bad factor. Suppose $N(a + \alpha b) = \Pi_i l_i^{e_i}$ with l_i prime. Then:*

(i) For all prime l not badly ramified and $r \in \mathbb{Z}/l\mathbb{Z}$ root of f, the ideal $\mathfrak{u}_0 := (\alpha - r)\mathcal{O} + l\mathcal{O}$ is prime and $N(\mathfrak{u}_\circ) = l$;

(ii) For all i, b is invertible modulo l_i. We can put $r_i = -b^{-1}a \bmod l_i$ and $\mathfrak{u}_i := (\alpha - r_i)\mathcal{O} + l_i\mathcal{O}$. Then $\mathfrak{u}_i$ is the unique ideal of $\mathcal{O}$ above both $(a - \alpha b)\mathcal{O}$ and l_i;

(iii) $(a - \alpha b)\mathcal{O} = \Pi_i \mathfrak{u}_i^{e_i}$.

proof:

(i) One easily checks that $[\mathbb{Z}[\alpha] : (\mathfrak{u}_0 \bigcap \mathbb{Z}[\alpha])] = l$. Call c the index $[\mathcal{O} : \mathbb{Z}[\alpha]]$. Since l is not badly ramified, we have $l \nmid c$ and therefore for all $x \in \mathcal{O}$ we have $[x - (c^{-1} \bmod l) \cdot c \cdot x] \in l\mathcal{O} \subset \mathfrak{u}_0$. Hence we obtain $[\mathcal{O} : \mathfrak{u}_0] = [\mathbb{Z}[\alpha] : (\mathfrak{u}_0 \bigcap \mathbb{Z}[\alpha])] = l$. Therefore $\mathfrak{u}_0$ is prime and has norm l.

(ii) Since $gcd(a, b) = 1$ and $N(a+\alpha b) = b^{deg(f)} f(-\frac{a}{b})$, we have $gcd(b, N(a+\alpha b)) = 1$ so r_i exists. Since $\alpha - r_i \in \mathfrak{u}_i$ we have $a + \alpha b = (-b) \cdot (\alpha - r_i) \in \mathfrak{u}_i$, so $\mathfrak{u}_i$ is above $a + \alpha b$.

Conversely, let $\mathfrak{u}$ be a prime ideal above $(a - \alpha b)\mathcal{O}$ and l_i. Since $l_i \nmid b$, there exists $b' \in \mathbb{Z}$ such that $b'b - 1 \in l_i\mathbb{Z} \subset \mathfrak{u}$. Since $(-b)(\alpha - r_i) = a + \alpha b \in \mathfrak{u}$ we obtain $\alpha - r_i \in \mathfrak{u}$. Therefore we have $\mathfrak{u}_i = (\alpha - r_i)\mathcal{O} + l_i\mathcal{O} \subset \mathfrak{u}$. Since $\mathfrak{u}_\mathfrak{i}$ is prime (maximal in Dedekind domain), $\mathfrak{u} = \mathfrak{u}_i$.

(iii) We have $N(a + \alpha b) = \Pi_i \Pi_{\mathfrak{u} \text{ above } (a+\alpha b)\mathcal{O} \text{ and } l_i\mathcal{O}} N(\mathfrak{u})^{e_\mathfrak{u}}$ for some integers $e_\mathfrak{u}$. By (i), $N(a + \alpha b) = \Pi_i N(\mathfrak{u}_\mathfrak{i})^{e_{\mathfrak{u}_i}}$. Since $[\mathcal{O} : \mathfrak{u}_i] = l_i$ we have $N(\mathfrak{u}_i) = l_i$ and we conclude $e_i = e_{\mathfrak{u}_i}$.□

Nevertheless, in practice we do not avoid bad primes because it would force us to loose all the twice-smooth numbers with bad factors. Think about the case when 2 is bad. Hence we use Buchmann-Lenstra exactly for those numbers with badly ramified primes.

Why is the complexity of NFS of type $L(\frac{1}{3})$? Number fields bring several obstructions which make the algorithm long, and its rigorous analysis complicated. Still, at this stage of the presentation we can foresee that there is an algorithm and even roughly estimate it's complexity. Indeed, the numbers we inspect for smoothness are of size p in Index Calculus and of size $\sqrt{p}$ in the Gaussian Integers. We will see that for the Number Fields we can expect $L_p(\frac{2}{3})$.

In the complexity analysis we need to know the cardinal of the factor bases. Call B the smoothness bound and suppose $deg(K_1), deg(K_2) = \mathcal{O}(log(B)^s)$ for some $s < 1$. Then the union of factor bases has less than B elements[11]. Indeed, there are $\pi(B) \sim B/\log(B)$ prime numbers less than B. Each prime number splits into at most $d_i := deg(K_i)$ prime ideals in K_i for $i = 1, 2$. In total $\frac{(d_1+d_2)B}{\log(B)}$ elements. Since $d_1, d_2 = \mathcal{O}(log(B)^s)$ we have $\frac{(d_1+d_2)B}{\log(B)} < B$.

The time spent by the algorithm is of type $Proba(smoothness) \cdot B + B^2$ by analogy with the Gaussian Integers. Let's compute parameters s, e and θ such that $B = L(s)$, $E = L(e)$ (the sieve bound) and $m = L(\theta)$ optimize the time. For all pair $(a, b) \in \mathbb{Z}^2$ and for $i = 1, 2$ we have $\mathrm{N}(a + b\alpha_i) = b^d f_i(-\frac{a}{b}) = \mathcal{O}(dL(s)^d |f_i|_\infty)$ where $|f_i|_\infty$ stands for the largest coefficient of f_i. We can guarantee $|f_i|_\infty \leq m$ by taking $f_1(X) = X - m$ and $f_2(X)$ the base-m-expansion of p, i.e. $d_2 = d = [log_m p]$ and $f_2(X) = a_d x^d + a_{d-1} X^{d-1} + \ldots + a_0$ with $p = a_d m^d + \ldots + a_0$. We can impose $a_d = 1$ by taking m close to $N^{\frac{1}{d}}$ but we do not need this in a first approximation. This choice of polynomials forces us to take $d = log(p)/log(m) = \frac{\log(p)^{1-\theta}}{(\log\log p)^{1-\theta}}$. Thus we have $\mathrm{N}(a + b\alpha_i) \leq L(max\langle e + 1 - \theta, \theta\rangle)$. We minimize the norm for $\theta = \frac{1+e}{2}$. We optimize the sieve if $E = P_{smooth}(L(\frac{e+1}{2}), L(s))^{-1}$ which imposes $e = 1 - 2s$. Hence the sieve takes time $E^2 = L(e)^2 = L(e) = L(1 - 2s)$. Since the linear algebra step takes time $B^2 = L(s)^2 = L(s)$ a trade-off sieve-linear algebra is $s = \frac{1}{3}$. This means that the number we inspect for smoothness, the product norm times absolute value, has size $L_p(\frac{2}{3})$ which is tiny compared to $\sqrt{p}$ for Gaussian Integers. Using equation 2.3 we obtain:

$$complexity(\mathrm{NFS}) = L(\frac{1}{3}).$$

2.8 Virtual Logarithms

2.8.1 Preparation

The obstructions of the NFS come from the difference between the output of the sieve and the input of the linear algebra step. Indeed, on the one hand the sieve

[11] In practice all our prime ideals are of degree one except those above bad primes. In average a polynomial of given degree has at most one root modulo each prime number, so a thumb rule for the number of prime ideals is $2 \cdot \pi(B)$.

finds systems of type:

$$\begin{cases} \delta\mathcal{O}_1 = \mathfrak{p}_1^{e_1} \dots \mathfrak{p}_k^{e_k} \\ \delta\mathcal{O}_2 = \mathfrak{q}_1^{f_1} \dots \mathfrak{q}_l^{f_l}. \end{cases}$$

On the other hand the linear algebra step needs linear equation of type

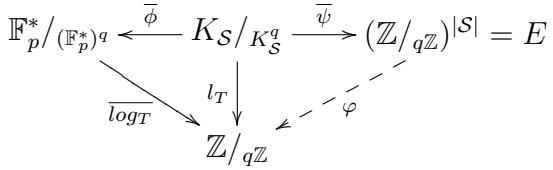
$$e_1 \text{“}log(\mathfrak{p}_1)'' + \dots + e_k \text{“}\log(\mathfrak{p}_k)'' = f_1 \text{“}\log(\mathfrak{q}_1)'' + \dots + f_l \text{“}\log(\mathfrak{q}_l)'' \bmod q$$

where $q|p-1$ and “$log(\mathfrak{p}_i)''$ is something that we do not know how to define.

There were two solutions found: one is to change what the linear algebra computes, the other is to give a deeper explanation for Index Calculus. In the first, one puts $K_1 = \mathbb{Q}$ and during the linear algebra step computes a vector $\langle x_{(a,b)} |\ (a,b)$ is double smooth$\rangle$ and an integer x such that $T^x S \Pi_{(a,b)} (a+b\alpha_1)^{x_{(a,b)}} = 1 \bmod p$ and $\Pi_{a,b}(a+\alpha_2 b)^{x(a,b)}$ is a q^{th} power. As a consequence $T^{-x} \equiv S \bmod p$. The disadvantage is that the linear algebra step depends on S, so the individual logarithm is as expensive as the precomputations. This is the case for both Gordon's [Gor93] and Schirokauer's [Sch93] algorithms.

The other solution is to see that the discrete logarithms of the factor base in Index Calculus are nothing but the coordinates of a vector $(l_1, \dots, l_R)$ with $R = \pi(B_1)$ and such that if S is B_1-smooth and if we know a factorization $S = p_1^{e_1} \dots p_R^{e_R}$ of S then $\log_T(S) = e_1 l_1 + \dots + e_R l_R$.

Let's be more precise. As usual p denotes the prime in input for DLP and q a prime divisor of $p-1$. Call $\mathcal{S} = \{p_1, \dots, p_R\}$ the factor base used in Index Calculus. In the diagram below $K_{\mathcal{S}} = \{p_1^{e_1} \dots p_R^{e_R} | e_1 \dots e_R \in \mathbb{Z}\}$. Call ϕ the projection of $\mathbb{Z}$ on $\mathbb{Z}/p\mathbb{Z}$ which induces a map $\phi|_{K_{\mathcal{S}}} : K_{\mathcal{S}} \to \mathbb{F}_p^*$. Put $\overline{\phi} : K_{\mathcal{S}}/K_{\mathcal{S}}^q \to \mathbb{F}_p^*/(\mathbb{F}_p^*)^q$ the obviously induced map. Put $\psi(x) = (val_{p_1}(|x|), \dots, val_{p_{|\mathcal{S}|}}(|x|))$ which gives an isomorphism $\psi : K_{\mathcal{S}} \to (\mathbb{Z}/q\mathbb{Z})^{|\mathcal{S}|}$ and call $E = (\mathbb{Z}/q\mathbb{Z})^{|\mathcal{S}|}$ which is a $\mathbb{F}_q$-vector space. Finally put $\overline{\psi} : K_{\mathcal{S}}/K_{\mathcal{S}}^q \to (\mathbb{Z}/q\mathbb{Z})^{|\mathcal{S}|}$ the obviously induced map.



Let's put $\varphi = l_t \circ \overline{\psi}^{-1}$ and notice that we have $\varphi \in E^*$. Now we understand Index Calculus in a different way: during the chose&check step we collect $|\mathcal{S}| + k$, $k = \mathcal{O}(1)$, pairs $(v_i, \varphi(v_i))_{1 \le i \le |\mathcal{S}|+k}$ and for heuristic reasons we expect the family $(v_i)_{1 \le i \le |\mathcal{S}|+k}$ to have rank $|\mathcal{S}|$. The linear algebra step is finding the matrix of φ from the values φ takes on a base. Finally the step 3 in Index Calculus is finding a relation between S and an element S' in K_S, computing the exponent vector

$e = \overline{\psi}(S')$ and obtaining $\overline{\log_T}(S') = \varphi(e)$. In short, steps 1 and step 2 find the horizontal vector in the next equation while step 3 finds the vertical one.

$$log_T(|S|) = \begin{pmatrix} log_T(2) & \dots & log_T(p_R) \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ \vdots \\ e_R \end{pmatrix}. \tag{2.7}$$

Notice that we compute the discrete log of $|S|$ instead of S since one can check that $log_T(-1) = \frac{p-1}{2}$. The case of -1 is explained below together with the other roots of unity.

2.8.2 Schirokauer's maps

Let q denote as usual a prime divisor of $p-1$ and K a number field (one of the two or more used in NFS). In the sequence we suppose $q^2 \nmid p-1$ and $q \nmid h_K$, the class number of K. Indeed, if $q \leq L_p(\frac{1}{3}, 2)$ then we use the Pollard's Rho method[12] instead of NFS, so we don't need Schirokauer's maps. If q is larger we have a probability[13] of at least $(1 - \frac{1}{q})$ that q does not divide the class group order of K. It is then reasonable in practice to forget about the case $q \mid h_K$. In theoretical study if this case occurs we run again the whole algorithm for a different polynomial f. It does not change the asymptotic complexity. A second difficulty comes if $q^e || p-1$ with $e \geq 2$. In this case the algorithms must be adjusted, but the modifications are short and they are described in detail in [Sch93].

Remember Dirichlet's Theorem which says that $\mathcal{O}_K^*$ is isomorphic to $\mu \times \mathbb{Z}^{r_\mathbb{R}+r_\mathbb{C}-1}$ where $(r_\mathbb{R}, r_\mathbb{C})$ is the signature[14] of K and μ is the group of roots of unity. If $gcd(\#\mu, q) = 1$, then[15] $\mathcal{O}^*/(\mathcal{O}^*)^q \simeq (\mathbb{Z}/q\mathbb{Z})^{r_\mathbb{R}+r_\mathbb{C}-1}$. As it comes to roots of order q, we avoid them by imposing $q \nmid disc(f)$ when we select f (in practice we ignore them). The lemma below makes it clear:

Lemma 1 *Let f be an irreducible polynomial in $\mathbb{Q}[X]$ and K its rupture field. If q is an odd prime number such that $q \nmid disc(f)$, then K has no roots of unity of order q.*

proof Suppose $\xi_q \in K$ is a q^{th} root of unity. Then $\mathbb{Q}(\xi_q) \subset K$. As a consequence all the prime numbers ramified over $\mathbb{Q}(\xi_q)$ are ramified over K. It is known that $|disc(\mathbb{Q}(\xi_q))|$ is a power of q, so q is ramified over $\mathbb{Q}(\xi_q)$, thus over K. Therefore

[12] Despite the bad complexity of Pollard's Rho, it is still fast enough. Indeed $\sqrt{q} \leq L(\frac{1}{3}, 2)^{\frac{1}{2}} = L(\frac{1}{3}, 1)$..

[13] See [CL84] for heuristics on the class group number.

[14] K has $r_\mathbb{R}$ embeddings in $\mathbb{R}$ and $2r_\mathbb{C}$ more in $\mathbb{C}$. Remember that $r_1 + 2r_2 = d$.

[15] One can check that the roots of unity are all q^{th} powers. It makes the condition Schirokauer [Sch05] puts when defining virtual logs superfluous.

q divides $disc(K)$ and since $disc(K)^2[\mathcal{O}_K : \mathbb{Z}[\alpha]] = \pm disc(f)$, we deduce that q divides $disc(f)$.$\square$

Now that we know there is an isomorphism $\mathcal{O}^*/(\mathcal{O}^*)^q \simeq (\mathbb{Z}/q\mathbb{Z})^r$ with $r := r_{\mathbb{R}} + r_{\mathbb{C}} \leq d$ we want to compute it. The naive idea is to map $\mathcal{O}^*$ in $\mathbb{R}^r$ by $x \mapsto (\log|\psi_1(x)|, \ldots, \log|\psi_r(x)|)$ where ψ_i are embeddings of K into $\mathbb{C}$. Unfortunately we do not know how to use this idea without executing the linear algebra step over $\mathbb{Z}$. If we do so even a sparse system like ours costs $\mathcal{O}(B^3)$ with B the size of the system and we obtain a poorer complexity. For example Gordon [Gor93] obtained $L_p(\frac{1}{3}, 2.080)$ for his discrete log by NFS. Instead we define r maps $(\lambda_1, \ldots, \lambda_r) : \mathcal{O}^* \to (\mathbb{Z}/q\mathbb{Z})^r$ such that the only x which verify $(\lambda_1(x), \ldots, \lambda_r(x)) = (0, \ldots, 0)$ are the q^{th} powers.

Definition 2 *Let q be a prime factor of $p-1$ which is unramified*[16] *over K and pick a base b of the free $\mathbb{Z}$-module $\mathcal{O}_K$. Let $f = \Pi_{i \leq k} f_i$ be the decomposition of f in $\mathbb{F}_q$ and $\epsilon = lcm\{deg f_i, 1 \leq i \leq k\}$. In the lemma below we show that $\lambda : \mathcal{O}_K^*/(\mathcal{O}_K^*)^q \to q\mathcal{O}_K/q^2\mathcal{O}_K$, $u \mapsto u^\epsilon - 1$ mod $q^2\mathcal{O}_K$ is a well defined homomorphism. We call maps of Schirokauer the coordinates $\lambda_1, \ldots, \lambda_d$ of λ in the base b.*

Notice that one computes the maps of Schirokauer in polynomial time. Also notice that we defined d maps of Schirokauer while we announced that r of them are enough. This is because the theorem below shows that the application $(\lambda_1, \ldots, \lambda_d) : \mathcal{O}_K^*/(\mathcal{O}_K^*)^q \to (\mathbb{Z}/q\mathbb{Z})^d$ has rank r and therefore one can select in polynomial time r indices $i_1, \ldots, i_r$ such that $ker(\lambda_1, \ldots, \lambda_d) = ker(\lambda_{i_1}, \ldots, \lambda_{i_r})$. So, the application $(\lambda_{i_1}, \ldots, \lambda_{i_r}) : \mathcal{O}_K^*/(\mathcal{O}_K^*)^q \to (\mathbb{Z}/q\mathbb{Z})^r$ is what we need.

Lemma 2 *The map $\lambda : \mathcal{O}_K^*/(\mathcal{O}_K^*)^q \simeq q\mathcal{O}_K/q^2\mathcal{O}_K$, $u \mapsto u^\epsilon - 1$ mod $q^2\mathcal{O}_K$ is a group homomorphism.*

proof: Let $\gamma, \gamma' \in \mathcal{O}_K^*$. For all $1 \leq i \leq k$, $\mathfrak{q}_i = q\mathcal{O}_K + (f_i)\mathcal{O}_K$ is prime and $\mathcal{O}_K/\mathfrak{q}_i \simeq \mathbb{F}_{q^{deg(f_i)}}$. Thus, for all $1 \leq i \leq k$ $\gamma^\epsilon \equiv 1 \bmod \mathfrak{q}_i$. Then, by Chinese Remainder Theorem, $\delta^\epsilon \equiv 1 \bmod \Pi_i \mathfrak{q}_i$, i.e. modulo $q\mathcal{O}_K$ because q is unramified. Therefore there exist $\delta \in \mathcal{O}_K$ such that $\gamma^\epsilon = 1 + q\delta$. Similarly there exist $\delta' \in \mathcal{O}_K$ such that $\gamma'^\epsilon = 1 + q\delta'$. Then we have: $(\gamma\gamma')^\epsilon = 1 + q(\delta + \delta') + q^2(\ldots)$. Therefore λ is a homomorphism.$\square$

The main theorem about Schirokauer's maps is as follows.

Theorem 3 *Let q be a prime factor of $p-1$ which is unramified over K. Suppose $q||p-1$, $q \nmid h_K$ and $[(\mathcal{O}_K)^* \bigcap (1 + q^2\mathcal{O}_K)] \subset (\mathcal{O}_K^*)^q$. Let $\gamma \in \mathcal{O}_K$ be such that:*

(i) $ord_{\mathfrak{p}}(\gamma) \equiv 0$ (mod $q\mathbb{Z}$) for all prime ideal $\mathfrak{p}$;

[16]The case q ramified makes no difference for the implementation, accepts a proof as long as in the unramified case, but it makes exposition less clear. For details see [Sch93]. If one wanted to avoid q to be ramified, one would only need to impose $q \nmid disc(f)$ when selecting f.

(ii) $\lambda(\gamma) = 0$.

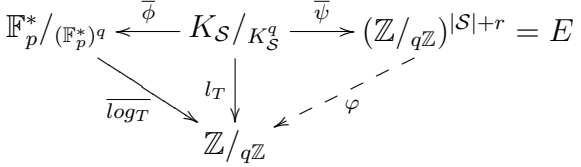
Then γ *is a* q^{th} *power.*

proof It is proposition 3.8 in [Sch93].

It means that, subject to the conditions of the theorem, Schirokauer's maps provide us with an isomorphism $\mathcal{O}^*/(\mathcal{O}^*)^q \to (\mathbb{Z}/q\mathbb{Z})^r$. The only condition which wasn't analyzed is the last one: $(\mathcal{O}_K)^* \bigcap (1 + q^2\mathcal{O}_K) \subset (\mathcal{O}_K^*)^q$. Unfortunately we do not have an equivalent condition easy to test. It makes algorithms using Schirokauer's maps heuristic. Nevertheless algorithms work in practice and Schirokauer [Sch93] proved that it comes to a matrix over $\mathbb{F}_q$ with seemingly random entries to be invertible.

2.8.3 Definition of the virtual logs

We draw once again the diagram in the preparation but we generalize the notations. Let K be K_i with $i = 1, 2$, $r = r_i$ the free rank of $\mathcal{O}_{K_i}^*$ and $\phi = \phi_i$ where ϕ_i is the function defined in section 2.7. Call $\mathcal{S}$ the set of prime ideals of K which we use as factor base. Put $K_{\mathcal{S}} = \{\gamma \in \mathcal{O}_K |$ all the ideal factors of $\gamma\mathcal{O}_K$ are in $\mathcal{S}\}$. Let $\phi : K_{\mathcal{S}} \to \mathbb{F}_p$ be the restriction of ϕ to $K_{\mathcal{S}}$. Put $\overline{\phi} : K_{\mathcal{S}}/K_{\mathcal{S}}^q \to \mathbb{F}_p^*/(\mathbb{F}_p^*)^q$ the obviously induced map. We put $\psi(x) = (val_{p_1}(x), \ldots, val_{p_{|\mathcal{S}|}}(x), \lambda_1(x), \ldots, \lambda_r(x))$ where the λ_j are Schirokauer's maps. We obtain an isomorphism $\psi : K_{\mathcal{S}} \to (\mathbb{Z}/q\mathbb{Z})^{|\mathcal{S}|+r}$. Call $E = (\mathbb{Z}/q\mathbb{Z})^{|\mathcal{S}|+r}$ which is a $\mathbb{F}_q$-vector space. Put $\overline{\psi} : K_{\mathcal{S}}/K_{\mathcal{S}}^q \to (\mathbb{Z}/q\mathbb{Z})^{|\mathcal{S}|+r}$ the induced map. Let log_T be the discrete logarithm and $\overline{\log_T}$ be $\log_T \mod q$. Finally put $l_T = \overline{\log_T} \circ \overline{\phi}$ which can be called the discrete logarithm modulo q of numbers in $\mathcal{O}_K$.



For $i = 1, 2$ we define φ_i as in the diagram. We can now define the following linear map: $\varphi : \mathbb{F}_q \times \mathbb{F}_q^{|S_1|+r_1} \times \mathbb{F}_q^{|S_2|+r_2} \to \mathbb{F}_q$, $\varphi(k, x, y) = k + \varphi_1(x_1, \ldots, x_{|\mathcal{S}_1|+r_1})) - \varphi_2(y_1, \ldots, y_{|\mathcal{S}_2|+r_2}))$. We know the value of $\varphi(1, 0, 0))$ which is 1. We also know $|S_1| + |S_2| + r_1 + r_2$ vectors in $ker(\varphi)$. One of them is $(1, \psi(x), 0)$ obtained from $\tau\mathcal{O}_{K_1} = \Pi\mathfrak{p}_i^{e_i(t)}$ where $\tau \in \mathcal{O}_{K_1}$ is such that $\phi_1(\tau) = T$. The others are of type $(0, \psi_1(a + \alpha_1 b), \psi_2(a + \alpha_2 b))$.

Definition 3 *The coordinates* $(\chi_{\mathfrak{p}_1}, \ldots, \chi_{\mathfrak{p}_{|\mathcal{S}|_1}}, \chi_1, \ldots, \chi_{r_1}, \chi_{\mathfrak{q}_1}, \ldots)$ *of* φ *are called virtual logarithms. In particular* $\chi_{\mathfrak{p}}$ *is the virtual log of* $\mathfrak{p}$ *and* χ_j *is the virtual log of the* j^{th} *map of Schirokauer.*

We call virtual logarithms like this because we compute them in the same way we used to compute the discrete logs of small primes in Index Calculus. Nevertheless we do not know any interpretation of them as logarithms. Notice that we have:

$$\forall s,\ log_t(s) = (\ \chi_{\mathfrak{p}_1} \cdots \chi_{\mathfrak{p}_{|S_1|}} | \chi_1 \cdots \chi_{r_1} | \chi'_{\mathfrak{q}_1} \cdots \chi'_{\mathfrak{q}_{|S_2|}} | \chi'_1 \cdots \chi'_{r_2}\ ) \cdot \begin{pmatrix} e_1(s) \\ \vdots \\ e_{|S_1|}(s) \\ \hline \lambda_1(s) \\ \vdots \\ \lambda_{r_1}(s) \\ \hline f_1(s) \\ \vdots \\ f_{|S_2|}(s) \\ \hline \lambda'_1(s) \\ \vdots \\ \lambda'_{r_2}(s) \end{pmatrix}. \quad (2.8)$$

In the past lines we stated without proof results which together make a demonstration of the following theorem.

Theorem 4 *Let K_1 and K_2 be two number fields without q^{th} roots of the unity. Define ψ_1 respectively ψ_2 like on the diagram above. Then we have:*

(i) For each $i = 1, 2$ and for each set of prime ideals $\mathcal{S}_i$ there is a unique vector φ_i which makes the diagram above commutative. It defines for each pair $(\mathcal{S}_1, \mathcal{S}_2)$ a linear form φ as above.

(ii) If $\mathcal{S}'_1 \subset \mathcal{S}_1$ is a different prime ideals set then φ'_1 is the restriction of φ_1 to a subspace. In particular if $\mathfrak{p} \in \mathcal{S}'_1 \subset \mathcal{S}_1$, then the coefficient $\chi_{\mathfrak{p}}$ of φ is equal to the coefficient $\chi'_{\mathfrak{p}}$ of φ'.

(iii) Suppose in addition we have a family $\{(a, b) | a, b \in \mathbb{Z}\}$ such that $\{(0, \psi_1(a + \alpha_1 b), \psi_2(a + \alpha_2 b))\}$ has rank $|\mathcal{S}_1| + |\mathcal{S}_2| + r_1 + r_2 - 1$, i.e. the sieve step managed to find a full rank family. Then the linear system $\{\varphi(1, \psi(x), 0) = 1, \forall (a, b), \varphi(0, \psi_1(a + \alpha_1 b), \psi_2(a + \alpha_2 b)) = 0\}$ has exactly one solution: the coefficients of φ.

It can be surprising that the definition of the virtual logarithms used the sets $\mathcal{S}_i$ since we proved later that the virtual logs are independent. The reason is that we could have given an equivalent definition like the one in proposition 3.4 in [Sch05]: $\chi_{\mathfrak{q}} \equiv [l_T(\theta) - \Sigma_{j=1}^{r} \chi_j \lambda_j(\theta)] \cdot h_K^{-1} \bmod q$ for any θ such that $\theta \mathcal{O}_K = \mathfrak{q}^{h_K}$. Nevertheless our definition is more algorithmic-oriented.

Notice that (i) is a rephrasing of theorem 3.2 in [Sch05]. In the same paper Schirokauer pointed out that a failure of ψ_1 or ψ_2 to be isomorphisms, i.e. a failure for the heuristics we made on Schirokauer's maps, cannot be detected until the whole algorithm has been run.[17] It should be also noted that the virtual logarithms depend on Schirokauer's maps which in turn depend on the bases of $\mathcal{O}_{K_i}$ we use. Therefore if one records the virtual logs for later individual log computations, one needs to record also the Schirokauer's maps used.

An important property of the virtual logarithms is (ii), especially the uniqueness of the coefficients χ_j corresponding to Schirokauer's maps. Indeed, in some algorithms like the descent by special Q we compute the coefficients χ_j using a small set $\mathcal{S}_1$ but we use equation 2.8 for numbers with prime ideal factors of large norm.

2.9 Factorization Factory[18]

Thanks to the Virtual logarithms the linear algebra step depends only on p which means we can re-use it for many individual logarithms. Unfortunately we cannot share the linear algebra step for several values of p since its result, the virtual logarithms, are strongly related to p. Nevertheless Coppersmith [Cop93] showed that we can share part of the sieving step. Indeed the algorithm works as soon as we have $m = L_p(\frac{2}{3}, \frac{1}{\delta})^{1+o(1)}$ which allows us to use the same m for all the primes p in $[p_0, p_0 \cdot L_{p_0}(\frac{2}{3}, \frac{1}{\delta})]$ with p_0 fixed. Since the sieve is easier, we can ask for smooth numbers of better quality, i.e. we can lower B. Therefore the matrix in the linear algebra step is smaller and we will obtain a complexity of $L_p(\frac{1}{3}, 1.639)$ for the precomputations. For now 18 years this algorithm has had only a theoretical purpose since it requires space $L_{p_0}(\frac{1}{3}, 1.639)$.

Let's make the computations. We use two number fields $K_1 = \mathbb{Q}$ and K_2 and a unique smoothness bound B. Let β and δ be parameters such that $B = [L_p(\frac{1}{3}, \beta)]$ and $d = [\delta(\log p)^{\frac{1}{3}}(\log\log p)^{-\frac{1}{3}}]$. Let ϵ be such that the permanent file records all pairs (a, b) with $|a|, |b| \leq L_{p_0}(\frac{1}{3}, \epsilon)$, $gcd(a, b) = 1$ and such that $a + bm_0$ is B-smooth. Notice that m_0 is the value of m computed for p_0 and is shared for all p such that $log_2(p_0) \leq log_2(p) \leq log_2(p_0) + \frac{1}{\delta}\log(p_0)^{2/3}\log\log(p_0)^{1/3}$. One checks that the probability P_2 for $(a + b\alpha)$ to be B-smooth is $L_p(\frac{1}{3}, \frac{\frac{1}{\delta}+\delta\epsilon}{3\beta})^{-1+o(1)}$. In the same way one computes P_1 the probability of $a + bm_0$ to be B-smooth $L_p(\frac{1}{3}, \frac{1}{3\delta\beta})^{-1+o(1)}$. The condition for the algorithm to succeed is that it finds enough relations, i.e.

[17] Conversely a success of the algorithm does not guaranty that the functions ψ_i are isomorphisms.

[18] The term "factorization factory" was coined by Coppersmith in [Cop93].

$E^2 P_2 P_1 \geq B$. This is:

$$2\epsilon - \frac{2}{3\delta\beta} - \frac{\epsilon\delta}{3\beta} \geq \beta. \tag{2.9}$$

The time of step 1 is BP_2^{-1} since the time to test smoothness is negligible. The time for step 2 is $L_p(\frac{1}{3}, 2\beta)^{1+o(1)}$ as in the other versions of NFS. Therefore we want to minimize:

$$max(2\beta, \beta + \frac{1}{3\beta\delta} + \frac{\epsilon\delta}{3\beta}). \tag{2.10}$$

We impose equality in 2.9 because a smaller sieving domain means a faster sieve and because equality means the smallest value for ϵ except if $\delta > 6\gamma$. The case $\delta > 6\gamma$ transforms our condition into a tautology which cannot be the case, so this case does not translate any practical situation. We also impose:

$$2\beta = \beta + \frac{1}{3\beta\delta} + \frac{\epsilon\delta}{3\beta}. \tag{2.11}$$

Indeed, when one minimizes $max(x, f(x))$ for some decreasing function f, on an open domain, one takes $x = f(x)$. Otherwise a small change of x would diminish the maximum in both cases $x > f(x)$ and $x < f(x)$. We did not prove that the time of the sieve is decreasing with the smoothness bound B, but we do not want to prove that we make the best choices of parameters. If one computes $2 \cdot (2.9) + (2.11)$, one obtains:

$$\epsilon = \frac{9\beta^2}{6\beta + \delta}. \tag{2.12}$$

Put $s = \frac{\epsilon}{\beta}$ and $t = \frac{\delta}{\beta}$. Then (2.12) becomes $s = \frac{9}{6+t}$ and the equality version of (2.9) becomes $\beta^3 = [s(2 - \frac{t}{3}) - 1]\frac{3}{2}t$. What we try to minimize is $2\beta = 2\sqrt[3]{h(t)}$ with $h(t) = \frac{6t(3-t)}{6+t}$. By derivation one finds that the optimal choice is $t = 3\sqrt{6} - 6$. Therefore $\beta = (\frac{5+2\sqrt{6}}{18})^{\frac{1}{3}} \approx 0.8193$, $\delta = \beta(3\sqrt{6} - 6) \approx 1.1048$ and $\epsilon = \beta \cdot \frac{\sqrt{6}}{2} \approx 1.0034$. This gives time $L_p(\frac{1}{3}, 2\beta)^{1+o(1)} = L_p(\frac{1}{3}, 1.639)^{1+o(1)}$ for steps 1 and 2 and $L_{p_0}(\frac{1}{3}, 2\epsilon)^{1+o(1)} = L_{p_0}(\frac{1}{3}, 2.007)$ for the pre-sieving procedure.

Remark 2 *We did not use the idea of multiple number fields here. Indeed using V fields multiplies the size of the linear system by V. Therefore we need to divide the smoothness bound by V. By computations one sees that the probability of a number to be B/V-smooth for one of V fields is smaller that the it's probability to be B-smooth for the first field. Indeed the probability is of type u^{-u} rather than $u^{-const.}$. This is not the first time we see the bad effect of multiple fields. In Coppersmith's modification to NFS we have a good idea-sharing the rational sieve- which should win us much of the time used for sieving. Nevertheless, the bad effect of multiple fields annihilated part of our gain so that we only have a speed-up from $L_p(\frac{1}{3}, 1.923)$ to $L_p(\frac{1}{3}, 1.902)$.*

2.10 Descent by Special Q

All the improvements previously presented speed up the precomputation stage. Therefore we need a new idea to speed up the individual logarithm from $L(\frac{1}{2})$ to $L(\frac{1}{3})$. We proceed as it follows. First pick h in the factor base and choose&check random z until $h^z S$ is $L_p(\frac{2}{3})$-smooth. Then write $h^z S = q_1 \dots q_2$ and compute the virtual logarithms of the q_i by a descent[19] as it follows. Let Q be the number of size $L_p(\frac{2}{3})$ we want to write in $\mathbb{F}_p$ as a product of numbers of size $L_p(\frac{1}{3})$. Choose a parameter $\nu \in (0,1)$ and sieve on the numbers of form $a + bm$, with $|a|, |b| \leq L_p(\frac{1}{3})Q^{\frac{1}{2}}$ and which are divisible by Q for those for which both $\frac{a+bm}{Q}$ and $a + b\alpha$ are Q^ν-smooth. We obtain a system:

$$\begin{cases} a + bm &= Q\Pi l^{e_{Q,l}} & l \leq Q^\nu, prime \\ (a + b\alpha)\mathcal{O}_{K_2} &= \Pi \mathfrak{u}^{m_{Q,\mathfrak{u}}} & N(\mathfrak{u}) \leq Q^\nu, \mathfrak{u} \text{ ideal in } K_2. \end{cases}$$

This gives us an equation:

$$\log_T Q \equiv \Sigma_{\mathfrak{u}} \chi_{\mathfrak{u}} m_{Q,\mathfrak{u}} + \Sigma_{k=1}^{r} \chi_k \lambda_k (a + b\alpha) - \Sigma_l \log_T l \bmod q \tag{2.13}$$

with $\chi_{\mathfrak{u}}$, χ_k and $\log_T l$ defined in 2.8.3. Then start again with each divisor l of $a+bm$ respectively each prime ideal divisor of $a+b\alpha$. Notice that we make descents for both numbers and ideal even if we started with a number.

The probability of a number $a+bm$ to be Q^ν-smooth is at least $P_{smooth}(L(\frac{2}{3}), L(\frac{1}{3}))$ which is $L(\frac{1}{3})^{-1}$. Next the probability that $a+b\alpha$ is Q^ν-smooth is $P_{smooth}(L(\frac{1}{3})Q^{\frac{1}{2}}, Q^\nu)$. This is the lowest when Q has size $L(\frac{1}{3})$ and it is $L(\frac{1}{3})^{-1}$. As a consequence we descend from the size of Q to the size of Q^ν after testing for smoothness $L(\frac{1}{3})$ pairs. One checks easily that the number of steps in the descent is $\mathcal{O}(e^{(\log\log p)^2})$, which gives the descent step a complexity of type $L(\frac{1}{3})$. See subsection 3.3.2 for full details.

[19] The idea of descent is present in [Cop02]. Joux and Lercier used in practice a particular case of descent without supplying theoretical analysis. They also suggested to perform it using Pollard's Lattice Sieve introduced in [Pol93]. Commeine and Semaev adapted the idea to NFS and proved that it has a complexity of $L(\frac{1}{3}, 1.442)$.

Chapter 3

New Algorithm: Discrete Logarithm Factory

3.1 The algorithm

The algorithm we are going to present combines all the ideas above and has complexity $L_p(\frac{1}{3}, 1.639)$ for the precomputations and $L_p(\frac{1}{3}, 1.232)$ for the individual logarithm. The algorithm uses a file of size $L_{p_0}(\frac{1}{3}, 1.639)$ computed after a presieving procedure which has complexity $L_{p_0}(\frac{1}{3}, 2.007)$.

Algorithm 2 *Discrete Logarithm Factory*

REQUIREMENTS: a prime p_0 *and the values for* $\delta = 1.1048$, $\beta = 0.8193$, $\epsilon = 1.0034$, $B = L_{p_0}(\frac{1}{3}, \beta)$, $E = L_{p_0}(\frac{1}{3}, \epsilon)$, $d = [\delta(\log p_0)^{\frac{1}{3}}(\log\log p_0)^{-\frac{1}{3}}]$, $c = 0.811$, $C = L_{p_0}(\frac{2}{3}, c)$, $\bar{c}_M = {c_M}^1 = 1$; $m_0 = [L_{p_0}(\frac{2}{3}, \frac{1}{\delta})]$; *a permanent file with the couples* (a, b) *such that* $a + m_0 b$ *is* B*-smooth.*

INPUT: p *of size* p_0 *and* $T, S \in \mathbb{F}_p^*$ *such that* T *spans* $\mathbb{F}_p^*$ *and* T *is* B*-smooth over* $\mathbb{Z}$*; a prime factor* q *of* $p - 1$*.*

OUTPUT:$log_T S$ *modulo* q*.*

0. *Select* $f \in \mathbb{Z}$ *irreducible such that* $p \mid f(m_0)$ *and* $|f|_\infty \leq m_0$*; set* $K_1 = \mathbb{Q}$, $K_2 = \mathbb{Q}[X]/\langle f \rangle$.

($\star$) *PRECOMPUTATIONS*

[1] c_M does not count in first approximation complexity. One has to find the best value in practice.

1.1 *SIEVE: Sieve on the polynomials* $\{a + bX \mid a, b$ *in the permanent file*$\}$ *until* B *pairs* (a, b) *are* B*-smooth* $\mathcal{O}_{K_2}$*;*

1.2 *LINEAR ALGEBRA: Compute the virtual logarithms by solving the linear system.*

($\star\star$) *INDIVIDUAL LOG*

2.1 *SMOOTHING: Take* h *a prime from the factor base over* $\mathbb{Q}$*; choose and check random* z *until* $h^z S$ *is* C*-smooth; factorize* $h^z S = q_1 q_2 \dots q_k$*;*

2.2 *DESCENT: For each* $j \leq k$ *do descent steps in order to express* $log_T q_j$ *with respect to the virtual logarithms computed in step* 1.2*.*

3.2 Early Abort Idea

3.2.1 One admissibility test

Smoothness Results We are given a generator of random numbers of size n and we need to find one which is $L_n(\frac{2}{3}, a)$-smooth for a value $a > 0$ we prefer. For the moment we fix a and we are going to chose it at the end of the analysis. Contrary to the naive idea where we test by ECM every number for $L_n(\frac{2}{3}, a)$-smoothness, we proceed in two steps. First we submit each number to a quick test called admissibility which discards most of the bad candidates and some of the good ones. The candidates who succeed the first test are called admissible and are submitted to the actual test of $L_n(\frac{2}{3}, a)$-smoothness. The numbers who succeed both tests are called admitted and are obviously good candidates. We show next that a good choice of the admissibility test allows us to asymptotically speed up the selection and that in the end we have at least one admitted candidate.

Let $0 < \theta, c < 1$ be parameters which will be chosen later. For all numbers $m \leq n$ we put m_1 the largest divisor of m which is $L_n(\frac{2}{3}, \theta a)$-smooth. Let's call M_{ble} the set of admissible candidates:

$$M_{ble} = \{m \leq n | \ \frac{m}{m_1} \leq n^{(1-c)}\}. \tag{3.1}$$

Next we define the set of admitted candidates:

$$M_{ed} = \{m \in M_{ble} | m \text{ is } L(\frac{2}{3}, a)\text{-smooth}\}. \tag{3.2}$$

In order to evaluate the cardinals of M_{ble} and M_{ed} we need the next theorem.

Notation 1 *For all* x, y, z *we put*

$$T(x, y, z) = \{m \leq x | \text{ for all prime divisors } p \text{ of } m, \ z \leq p \leq y\}.$$

We put $\psi(x, y, z) = \#T(x, y, z)$ *and* $P_{smooth}(x, y, z) = \psi(x, y, z)/x$*.*

Theorem 5 *Let $\epsilon > 0$ be arbitrary. If $u = \frac{\log x}{\log y}$ and if $z < y^{1-\frac{1}{\log u}}$, then*

$$\psi(x, y, z) = x \cdot u^{-u+o(u)} \tag{3.3}$$

uniformly for $(\log x)^{1+\epsilon} < y < exp(\log x)^{1-\epsilon}$, $x \geq 10$.

proof: It is theorem 2.2. in [Pom82]. The proof uses theorem [CEP83].

Remark 3 *Since the function $u \mapsto u^{-u}$ is decreasing we have $P_{smooth}(x, y, z) \geq P_{smooth}(x', y, z)$ as soon as $x \leq x'$.*

A convenient corollary translates the theorem in the L notation:

Corollary 2 *Let $n \in \mathbb{N}$. Let $x = L_n(s_x, c_x)$, $y = L_n(s_y, c_y)$ and $z = L_n(s_z, c_z)$ such that $0 < s_y < s_x$ and $(s_z, c_z) < (s_y, c_y)$ in lexico-graphical order. Then $\psi(x, y, z) = x \cdot L_n(s_x - s_y, -(s_x - s_y)\frac{c_x}{c_y})^{1+o(1)}$.*

proof: The condition $0 < s_y < s_x$ guarantees that we have $(\log x)^{1+\epsilon} < y < exp(\log x)^{1-\epsilon}$, while $(s_z, c_z) < (s_y, c_y)$ guarantees that $z < y^{1-\frac{1}{\log u}}$. We have $u = L_n(s_x - s_y, \frac{c_x}{c_y})$ and $\log u = \frac{c_x}{c_y}(\log n)^{s_x - s_y}(\log\log n)^{1-s_x+s_y}$. Finally $\psi(x, y, z) = x \cdot e^{-u(\log u)(1+o(1))} = L_n(s_x - s_y, (s_x - s_y)\frac{c_x}{c_y})^{1+o(1)}$.□

Now we have the main result:

Theorem 6 *Let $0 < c, \theta < 1$ and $n \in \mathbb{N}$ and define M_{ble} and M_{ed} as above. Then:*

(i) $\#M_{ble} \leq n \cdot L_n(\frac{1}{3}, -\frac{c}{3\theta a})^{1+o(1)}$;

(ii) $\#M_{ed} \geq n \cdot L_n(\frac{1}{3}, -\frac{c}{3\theta a} - \frac{1-c}{3a})^{1+o(1)}$.

proof:Let's call $L = L_n(\frac{2}{3}, 1)$.

(i) By partitioning M_{ble} on the non smooth part of each element we have:

$$\#M_{ble} = \Sigma_{t \in T(n^{1-c}, n^{1-c}, L^{\theta a})} \psi(\frac{n}{t}, L^{\theta a}, 1) \tag{3.4}$$

$$= \Sigma_{t \in T(n^{1-c}, n^{1-c}, L^{\theta a})} \frac{n}{t} P_{smooth}(\frac{n}{t}, L^{\theta a}, 1) \tag{3.5}$$

Using remark 3 for $x' = \frac{n}{t}$ and $x = \frac{n}{n^{1-c}}$ we have:

$$\#M_{ble} \leq \Sigma_{t \in T(n^{1-c}, n^{1-c}, L^{\theta a})} \frac{n}{t} P_{smooth}(\frac{n}{n^{1-c}}, L^{\theta a}, 1). \tag{3.6}$$

The right member of the inequality becomes by the corollary:

$$= n \cdot L_n(\frac{1}{3}, -\frac{c}{3\theta a})^{1+o(1)} \cdot \Sigma_{t \in T(n^{1-c}, n^{1-c}, L^{\theta a})} \frac{1}{t}. \tag{3.7}$$

Since $T(n^{1-c}, n^{1-c}, L^{\theta a}) \subset [1, n^{1-c}] \bigcap \mathbb{N} \subset [[1, n]]$ and $\Sigma_{t\in[[1,n]]} \frac{1}{t} \leq 1 + \log n$ we get:

$$\#M_{ble} \leq n \cdot L_n(\frac{1}{3}, -\frac{c}{3\theta a})^{1+o(1)} \cdot \log n = n \cdot L_n(\frac{1}{3}, -\frac{c}{3\theta a})^{1+o(1)}. \tag{3.8}$$

(ii) We have $M_{ed} = \{m_1 \cdot t \mid m_1 t \leq n,\ m_1 \in T(n, L^{\theta a}, 1),\ t \in T(n^{1-c}, L^a, L^{\theta a})\} = \Sigma_{t\in T(n^{1-c},L^a,L^{\theta a})} \#T(\frac{n}{t}, L^{\theta a}, 1)$. For $t \leq n^{1-c}$ we have $\frac{n}{t} \geq n^c$ and therefore $\#T(\frac{n}{t}, L^{\theta a}, 1) \geq \#T(n^c, L^{\theta a}, 1)$. Hence we obtain $\#M_{ed} \geq \psi(n^{1-c}, L^a, L^{\theta a}) \cdot \psi(n^c, L^{\theta a}, 1)$. A direct application of corollary 2 gives:

$$\#M_{ed} \geq n^c L_n(\frac{1}{3}, -\frac{c}{3\theta a})^{1+o(1)} n^{(1-c)} L_n(\frac{1}{3}, -\frac{1-c}{3a})^{1+o(1)} =$$

$$= n \cdot L_n(\frac{1}{3}, -\frac{c}{3\theta a} - \frac{1-c}{3\ a})^{1+o(1)}.\square$$

Time Analysis The condition for the early abort selection to find an admitted candidate is that it considers at least $\frac{n}{\#M_{ed}}$ candidates. Inspecting for $L_p(\frac{1}{3}, a\theta)$-smoothness this number of candidates takes time $\frac{n}{\#M_{ed}} \cdot t_{ECM}(L_n(\frac{2}{3}, a\theta))$, where t_{ECM} is the time of a test ECM. For all the admissible candidates we apply the actual test which takes time $t_{ECM}(L_n(\frac{2}{3}, a))$. The number of admissible candidates is $\frac{n}{\#M_{ed}} \cdot Prob(m \in M_{ble} | m \leq n) = \frac{n}{\#M_{ed}} \cdot \frac{\#M_{ble}}{n} = \frac{\#M_{ble}}{\#M_{ed}}$. Thus the total time is

$$\frac{n}{\#M_{ed}} \cdot t_{ECM}(L_n(\frac{2}{3}, a\theta)) + \frac{\#M_{ble}}{\#M_{ed}} \cdot t_{ECM}(L_n(\frac{2}{3}, a)) \tag{3.9}$$

which according to theorem 6 is less than

$$L_n(\frac{1}{3}, \frac{1-c}{3a} + \frac{c}{3\theta a} + \sqrt{\frac{4}{3}\theta a}) + L_n(\frac{1}{3}, \frac{1-c}{3a} + \sqrt{\frac{4}{3}a}). \tag{3.10}$$

For fixed a and c we minimize the time for $\theta = (\frac{c^2}{3a^3})^{\frac{1}{3}}$. We obtain the time:

$$L_n(\frac{1}{3}, \frac{1-c}{3a} + 3(\frac{c}{9})^{\frac{1}{3}}) + L_n(\frac{1}{3}, \frac{1-c}{3a} + \sqrt{\frac{4}{3}a}). \tag{3.11}$$

We minimize for $c = \frac{(\frac{4}{3}a)^{\frac{3}{2}}}{3}$. We obtain a function in a which reaches its minimum for $a = 0.7715$. We obtain:

$$\text{time(selection)}{=}L_n(\tfrac{1}{3}, 1.296)^{1+o(1)}.$$

3.2.2 Early Abort Strategy

One can improve the time by making several admissibility tests in a row. Take $k \in \mathbb{N}$. Let $0 < \theta_1 < \ldots < \theta_k < 1$ and $0 < c_1, \ldots, c_k < 1$ be parameters such that $c_1 + \ldots + c_k < 1$. For all $m \in [[1, n]]$ call m_i the largest divisor of m in $\psi(n, L_n(2/3, \theta_i a), L_n(2/3, \theta_{i-1} a))$. Put $M_{ble}^0 = [[1, n]]$. For $1 \leq i \leq k$ we call

$$M_{ble}^i = \{m \in M_{ble}^{i-1} |\ m_i \geq n^{c_i}\}.$$

Also put:

$$M_{ed} = \{m \in M_{ble}^k | m \text{ is } L_n(\frac{2}{3}, a)\text{-smooth}\}. \tag{3.12}$$

A similar theorem to 6 holds.

Theorem 7 *With the notations above we have:*

(i) $\#M_{ble}^i \leq n \cdot L_n(\frac{1}{3}, -\frac{c_1}{3\theta_1 a} - \ldots - \frac{c_i}{3\theta_i a})^{1+o(1)}$;

(ii) $\#M_{ed} \geq n \cdot L_n(\frac{1}{3}, -\frac{c_1}{3\theta_1 a} - \ldots - \frac{c_k}{3\theta_k a} - \frac{1-c_1-\ldots-c_k}{3a})^{1+o(1)}$.

proof: We put $L = L_n(\frac{2}{3}, 1)$.

(i) Take $N = [\log(n)^2]$. Put $\theta_0 = 0$.
For all $1 \leq l \leq k$ put $T^{(l)} = T(n, L^{\theta_l a}, L^{\theta_{l-1} a}) \bigcap [n^{c_l}, n]$,
$C^{(l)} = T(n^{1-c_1-\ldots-c_l}, n^{1-c_1-\ldots-c_l}, L^{\theta_l a})$, $\epsilon_l = \frac{1-c_l}{N}$ and $\eta_l = \frac{1-c_1-\ldots-c_l}{N}$. For all $l \leq k$ and $0 \leq i_l \leq N-1$ put

$$T_i^{(l)} = T^{(l)} \bigcap [n^{c_l + i\epsilon_l}, n^{c_l + (i+1)\epsilon_l}]$$

and

$$C_i^{(l)} = C^{(l)} \bigcap [n^{1-c_l-\ldots-c_l-(i+1)\eta_l}, n^{1-c_l-\ldots-c_l-i\eta_l}].$$

Let $1 \leq l \leq k$. Since each number is uniquely determined by its largest divisors in $T^{(l')}$ for $l' = 1, k$, we write all elements of $M_{ble}^{(l)}$ as $m = t_1 \cdot \ldots \cdot t_l t$ with $t_{l'} \in T^{(l')}$ for $l' < l$, $t \in C^{(l)}$ and $t_l \in \psi(\frac{n}{t_1 \ldots t_{l-1} t}, L^{\theta_l a}, L^{\theta_{l-1} a})$. Thus we have:

$$\#M_{ble}^{(l)} \leq \Sigma_{t \in C^{(l)}} \Sigma_{t_1 \in T^{(1)}} \ldots \Sigma_{t_{l-1} \in T^{(l-1)}} \psi(\frac{n}{t_1 \ldots t_{l-1} t}, L^{\theta_l a}, L^{\theta_{l-1} a}).$$

We need to be more precise, so we use $T^{(l)} = \bigcup_{i=1}^k T_i^{(l)}$ and $C^{(l)} = \bigcup_{i=1}^k C_i^{(l)}$. It gives:

$$= \Sigma_{0 \leq i, i_1, \ldots, i_{l-1} \leq N-1} \Sigma_{t \in C_i^{(l)}} \Sigma_{t_1 \in T_{i_1}^{(1)}} \ldots \Sigma_{t_{l-1} \in T_{i_{l-1}}^{(l-1)}} \psi(\frac{n}{t_1 t_2 \ldots t_{l-1} t}, L^{\theta_l a}, L^{\theta_{l-1} a}).$$

We use $n^{c_{l'}+i_{l'}\epsilon_{l'}}$ as a lower bound for $t_{l'} \in T_{i_{l'}}^{(l')}$ and we obtain:

$$\#M_{ble}^{(l)} \leq \Sigma_{i,i_1,\ldots,i_{l-1}} \Sigma_{t\in C_i^{(l)}} \Sigma_{t_1\in T_{i_1}^{(1)}} \ldots \Sigma_{t_{l-1}\in T_{i_{l-1}}^{(l-1)}}$$

$$\psi(n^{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} + (i+1)\eta_l}, L^{\theta_l a}, L^{\theta_{l-1} a}).$$

We factorize the common part for the terms in the sum and find:

$$\leq \Sigma_{i,i_1,\ldots,i_{l-1}} \#C_i^{(l)} \#T_{i_1}^{(1)} \ldots \#T_{i_{l-1}}^{(l-1)} \psi(n^{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} + (i+1)\eta_l}, L^{\theta_l a}, L^{\theta_{l-1}}).$$

For each of $\#C_i^{(l)}$, $\#T_{i_1}^{(1)}$, $\ldots$ and $\#T_{i_{l-1}}^{(l-1)}$ we have an upper bound in terms of $\psi(x, y, z)$:

$$\#M_{ble}^{(l)} \leq \Sigma_{i,i_1,\ldots,i_{l-1}} \tag{3.13}$$

$$\psi(n^{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} + (i+1)\eta_l}, L^{\theta_l a}, L^{\theta_{l-1} a}) \cdot \psi(n^{1-c_1-\ldots-c_l-i\eta_l}, n^{1-c_1-\ldots-c_l-i\eta_l}, L^{\theta_l a}) \cdot$$

$$\cdot \psi(n^{c_1+(i_1+1)\epsilon_1}, L^{\theta_1 a}, L^{\theta_0 a}) \ldots \psi(n^{c_l+(i_l+1)\epsilon_l}, L^{\theta_l a}, L^{\theta_{l-1} a})$$

which further gives:

$$\leq \Sigma_{i,i_1,\ldots,i_{l-1}} \psi(n^{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} + (i+1)\eta_l}, L^{\theta_l a}) \cdot \psi(n^{1-c_1-\ldots-c_l-i\eta_l}, n^{1-c_1-\ldots-c_l-i\eta_l}) \cdot$$

$$\cdot \psi(n^{c_1+(i_1+1)\epsilon_1}, L^{\theta_1 a}) \ldots \psi(n^{c_{l-1}+(i_{l-1}+1)\epsilon_{l-1}}, L^{\theta_{l-1} a}).$$

The product $\psi(n^{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} + (i+1)\eta_l}, L^{\theta_l a}) \cdot \psi(n^{1-c_1-\ldots-c_l-i\eta_l}, n^{1-c_1-\ldots-c_l-i\eta_l})$ decreases when i is larger because we are asking a bigger part of an element of size $n^{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} + (i+1)\eta_l + 1 - c_1 - \ldots - c_l - i\eta_l}$ to be $L^{\theta_l a}$-smooth. Hence we have an upper bound for $i = 0$. Therefore $\#M_{ble}^{(l)}$ is less than:

$$N\Sigma_{i_1,\ldots,i_{l-1}} \psi(n^{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} + \eta_l}, L^{\theta_l a}) \cdot \psi(n^{1-c_1-\ldots-c_l}, n^{1-c_1-\ldots-c_l}) \cdot \tag{3.14}$$

$$\cdot \psi(n^{c_1+(i_1+1)\epsilon_1}, L^{\theta_1 a}) \ldots \psi(n^{c_l+(i_l+1)\epsilon_l}, L^{\theta_l a}).$$

Put $\tau = \frac{\theta_{l-1}}{\theta_l} = \frac{max\{\theta_j | j \leq l-1\}}{\theta_l}$. Since $0 < \theta_1 < \ldots < \theta_k$ we have $0 < \tau < 1$. Call $f(i_1, \ldots, i_l)$ the term of the sum in equation (3.14). We cut the sum depending on the condition $\epsilon_1 i_1 + \ldots + \epsilon_{l-1} i_{l-1} > \tau c_l$ and we write:

$$\#M_{ble}^{(l)} \leq N[\Sigma_{\substack{0\leq i_1,\ldots,i_l\leq N-1 \\ \epsilon_1 i_1 + \ldots + \epsilon_{l-1} i_{l-1} > \tau c_l}} f(i_1, \ldots, i_{l-1}) + \Sigma_{\substack{0\leq i_1,\ldots,i_{l-1}\leq N-1 \\ \epsilon_1 i_1 + \ldots + \epsilon_{l-1} i_{l-1} \leq \tau c_l}} f(i_1, \ldots, i_{l-1})].$$

Let $0 \leq i_1, \ldots, i_{l-1} \leq N - 1$ be such that $\epsilon_1 i_1 + \ldots + \epsilon_{l-1} i_{l-1} > \tau c_l$. We divide the inequality by $max\{\theta_j | j \leq l-1\}$, we increase the larger member and we obtain:

$$\frac{1}{max\{\theta_j | j < l\}} [(i_1 + 1)\epsilon_1 + \ldots + (i_{l-1} + 1)\epsilon_{l-1}] > \frac{c_l}{\theta_l}$$

which we transform by using $max\{\theta_j | j < l\} \geq \theta_j$ for all $j < l$ and by dividing by $3a$:

$$\frac{(i_1+1)\epsilon_1}{\theta_1} + \ldots + \frac{(i_{l-1}+1)\epsilon_{l-1}}{\theta_{l-1}} > \frac{c_l}{\theta_l}$$

$$\frac{c_1+(i_1+1)\epsilon_1}{3\theta_1 a} + \ldots + \frac{(c_{l-1}+i_{l-1}+1)\epsilon_{l-1}}{3\theta_{l-1}a} > \frac{c_1}{3\theta_1 a} + \ldots + \frac{c_{l-1}}{3\theta_{l-1}a} + \frac{c_l}{3\theta_l a}.$$

It implies:

$$L_n(\frac{1}{3}, -\frac{c_1+(i_1+1)\epsilon_1}{\theta_1 a})^{1+o(1)} \cdot \ldots \cdot L_n(\frac{1}{3}, -\frac{c_{l-1}+(i_{l-1}+1)\epsilon_1}{\theta_{l-1}a})^{1+o(1)} \leq$$

$$\leq L_n(\frac{1}{3}, -\frac{c_1}{3\theta_1 a} - \ldots - \frac{c_l}{3\theta_l a})^{1+o(1)}$$

which we combine with the estimations for $1 \leq l' < l$,

$$\psi(n^{c_{l'}+(i_{l'}+1)\epsilon_{l'}}, L_n(\frac{2}{3}, \theta_{l'}a)) = n^{c_{l'}+(i_{l'}+1)\epsilon_{l'}} \cdot L_n(\frac{1}{3}, -\frac{c_{l'}+(i_{l'}+1)\epsilon_{l'}}{3\theta_{l'}a})^{1+o(1)}$$

in order to obtain:

$$\psi(n^{c_1+(i_1+1)\epsilon_1}, L^{\theta_1 a}) \ldots \psi(n^{c_{l-1}+(i_{l-1}+1)\epsilon_{l-1}}, L^{\theta_{l-1}a}) \leq n^{\sigma} L(\frac{1}{3}, -\frac{c_1}{3\theta_1 a} - \ldots - \frac{c_l}{3\theta_l a})^{1+o(1)}$$

with $\sigma = c_1 + \ldots + c_{l-1} + (i_1+1)\epsilon_1 + \ldots + (i_{l-1}+1)\epsilon_{l-1}$.
Since $\psi(n^{1-c_1-\ldots-c_l}, n^{1-c_1-\ldots-c_l}) = n^{1-c_1-\ldots-c_l}$ and $\psi(n^{c_l-i_1\epsilon_1-\ldots-i_{l-1}\epsilon_{l-1}+\eta_l}, L^{\theta_l a}) \leq n^{c_l-i_1\epsilon_1-\ldots-i_{l-1}\epsilon_{l-1}+\eta_l}$ in equation (3.14) we have

$$f(i_1, \ldots, i_{l-1}) \leq n^{1-c_1-\ldots-c_{l-1}-i_1\epsilon_1-\ldots-i_{l-1}\epsilon_{l-1}+\eta_l}.$$

$$\psi(n^{c_1+(i_1+1)\epsilon_1}, L^{\theta_1 a}) \ldots \psi(n^{c_{l-1}+(i_{l-1}+1)\epsilon_{l-1}}, L^{\theta_{l-1}a})$$

and we obtain:

$$f(i_1, \ldots, i_{l-1}) \leq n^{\epsilon_1+\ldots+\epsilon_{l-1}+\eta_l} n \cdot \tag{3.15}$$

$$\cdot L(\frac{1}{3}, -\frac{c_1}{3\theta_1 a} - \ldots - \frac{c_l}{3\theta_l a})^{1+o(1)}.$$

Let now $0 \leq i_1, \ldots, i_{l-1} \leq N-1$ such that $\epsilon_1 i_1 + \ldots + \epsilon_{l-1} i_{l-1} \leq \tau c_l$. Since $\tau < 1$ we have $c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1} \geq (1-\tau)c_l > 0$ and we can apply corollary 2 for all ψ expressions in $f(i_1, \ldots, i_{l-1})$. We obtain:

$$f(i_1, \ldots, i_{l-1}) \leq n^{\epsilon_1+\ldots+\epsilon_{l-1}+\eta_l}.$$

$$\cdot n \cdot L(\frac{1}{3}, -\frac{c_1+i_1\epsilon_1}{3\theta_1 a} - \ldots - \frac{c_{l-1}+i_{l-1}\epsilon_{l-1}}{3\theta_{l-1}a} - \frac{c_l - i_1\epsilon_1 - \ldots - i_{l-1}\epsilon_{l-1}}{3\theta_l a})^{1+o(1)}.$$

We minimize $\frac{c_1+i_1\epsilon_1}{3\theta_1 a}+\ldots+\frac{c_{l-1}+i_{l-1}\epsilon_{l-1}}{3\theta_{l-1}a}+\frac{c_l-i_1\epsilon_1-\ldots-i_{l-1}\epsilon_{l-1}}{3\theta_l a}$ for $i_1=\ldots=i_{l-1}=0$. Thus we have:

$$f(i_1,\ldots,i_{l-1})\leq n\cdot L(\frac{1}{3},-\frac{c_1}{3\theta_1 a}-\ldots-\frac{c_l}{3\theta_l a})^{1+o(1)}\cdot n^{\epsilon_1+\ldots+\epsilon_{l-1}+\eta_l}. \quad (3.16)$$

When we use (3.15) and (3.16) in (3.14) we have:

$$\#M_{ble}^{(l)}\leq (N^l n^{\epsilon_1+\ldots+\epsilon_{l-1}+\eta_l})\cdot n\cdot L(\frac{1}{3},-\frac{c_1}{3\theta_1 a}-\ldots-\frac{c_l}{3\theta_l a})^{1+o(1)}.$$

Since $N=[\log(n)^2]$, $N^l=L^{o(1)}$ and $n^{\epsilon_1+\ldots+\epsilon_{l-1}+\eta_l}=\exp(\frac{[l-2(c_1+\ldots+c_{l-1})-c_l]\log(n)}{N})=\exp(o(1))=\mathcal{O}(1)$. We conclude:

$$\#M_{ble}^{(l)}\leq n\cdot L_n(\frac{1}{3},-\frac{c_1}{3\theta_1 a}-\ldots-\frac{c_l}{3\theta_l a})^{1+o(1)}.$$

(ii) For $1\leq i\leq k$, put $S_i=[n^{c_i},n^{c_i}L^{\theta_i a}]\bigcap T(n^{c_i}L^{\theta_i a},L^{\theta_i a},L^{\theta_{i-1}a})$. The map $(m_1,\ldots,m_k,t)\mapsto t_1\ldots t_k t$ is an injection from

$$S_1\times\ldots\times S_k\times T(n^{1-c_1-\ldots-c_k}L^{-(\theta_1+\ldots+\theta_k)a},L^a,L^{\theta_k a}))\text{ to }M_{ed}.$$

Since the map $(m,\pi)\mapsto m\cdot\pi$ from $T(n_i^c,L^{\theta_i a},L^{\theta_{i-1}a})\times(\Pi(L^{\theta_i a})-\Pi(L^{\theta_{i-1}a}))$ can take at most $\frac{\log(n)}{\log(L^{\theta_i a})}$ times the same value, we have $\frac{\psi(n^{c_i}\cdot L^{\theta_i a},L^{\theta_i a},L^{\theta_{i-1}a})}{\psi(n^{c_i},L^{\theta_i a},L^{\theta_{i-1}a})}$ goes to infinity an in particular is larger than 2. Therefore, $\#S_i\geq(2-1)\psi(n^{c_i},L^{\theta_i a},L^{\theta_{i-1}a})$. A direct use of corollary 2 gives $\#M_{ed}\geq nL_n(\frac{1}{3},-\frac{c_1}{3\theta_1 a}-\ldots-\frac{c_k}{3\theta_k a}-\frac{1-c_1-\ldots-c_k}{3a})^{1+o(1)}$. □

Time Analysis We can now write the total time:

$$time=\frac{\#M_{ble}^0}{\#M_{ed}}t_{ECM}(L_n(\frac{2}{3},\theta_1 a))+\ldots+\frac{\#M_{ble}^{k-1}}{\#M_{ed}}t_{ECM}(L_n(\frac{2}{3},\theta_k a))+\frac{\#M_{ble}^k}{\#M_{ed}}t_{ECM}(L_n(\frac{2}{3},a)). \quad (3.17)$$

Therefore $time=L_n(\frac{1}{3},max\{f_1,\ldots,f_{k+1}\})$ where $f_i=\frac{c_i}{3\theta_i a}+\ldots+\frac{c_k}{3\theta_k a}+\frac{1-c_1-\ldots-c_k}{3a}+\sqrt{\frac{4}{3}\theta_i a}$ for $i=1,k$ and $f_{k+1}=\frac{1-c_1-\ldots-c_k}{3a}+\sqrt{\frac{4}{3}a}$. We put $\theta_{k+1}=1$ since it unifies the definitions for f_i with $i=1,k$ and for f_{k+1}. For $\theta_{i+1},\ldots,\theta_{k+1},c_{i+1},\ldots,c_k$ and c_i fixed we minimize f_i by imposing a relation between c_i and θ_i:

$$\theta_i=(\frac{c_i}{3a^3})^{\frac{1}{3}}.$$

For each $i\leq k$ as soon as $\theta_{i+1},\ldots,\theta_{k+1},c_{i+1},\ldots,c_k$ are fixed and (θ_i,c_i) satisfy the preceding relation we impose $f_i=f_{i+1}$ by

$$c_i=9(\frac{\omega}{3})^3\text{ with }\omega=(\frac{4}{3}a\theta_{i+1})^{\frac{1}{2}}.$$

This allows us to compute in a row c_k, θ_k, ..., c_1, θ_1. Hence we obtain the time for each value of a. Starting from the value of a in the case $k = 1$ we slowly change a in order to minimize the time. A good choice is $a = 0.811$ and $k = 8$. In this case we have

$$time(\text{strategy}) = L_n(\frac{1}{3}, 2.232)^{1+o(1)}.$$

3.3 Complexity Analysis of the Discrete Logarithm Factory

In section "Factorization Factory" we computed the complexity for steps SIEVE and LINEAR ALGEBRA. Let's see in detail smoothing and DESCENT.

3.3.1 Smoothing

Let C denote the smoothness bound in SMOOTHING. In order to prove that the choice in Discrete Logarithm Factory is optimal we consider the parameter $\theta > 0$ such that $C = L_p(\theta)$. An ECM test costs $L_p(\frac{\theta}{2})$ as pointed out in the smoothness section. Therefore the total time of the smoothness step is:

$$P_{smooth}(L_p(1), L_p(\theta))^{-1} \cdot L_p(\frac{\theta}{2}). \tag{3.18}$$

This is minimal for $\theta = \frac{2}{3}$. Therefore we have to take as smoothness bound $C = L_p(\frac{2}{3}, a)$ with a parameter. According to subsection 3.2.2 a good choice is $a = 0.811$. We obtain:

$$time(smoothing) = L_p(\frac{1}{3}, 1.232)^{1+o(1)}.$$

Remark 4 *The smoothing step does not depend on the other steps and we will check that it is the dominant step in the individual logarithm part.*

3.3.2 Descent

As announced in 2.10 we provide the full details for the descent by special Q. We reduce a number Q of size $L_p(\frac{2}{3}, c)$ with c a parameter chose at the smoothing step, for example $c = 0.811$ for the early abort strategy. Hence the complexity of the descent depends on c. Other global parameters which affect the time of the descent are β, such that the rational smoothness bound is $B_1 = L_p(\frac{1}{3}, \beta)$, γ, such that B_2 the algebraic bound is $B_2 = L_p(\frac{1}{3}, \gamma)$ and the number V of number fields we use on the algebraic side. This is 1 in the case of the Discrete Logarithm

Factory, but V with $V = [L_p(\frac{1}{3}, \beta - \gamma)]$ for the algorithm of Commeine and Semaev that we are going to speed up. Also remember that m is a root if f in $\mathbb{F}_p$ and that α_j is s complex root of $f + j(X - m)$ in Commeine and Semaev's algorithm.

The algorithm uses different procedures depending on the nature of the input: a number or an ideal. Each procedure chooses its parameters depending on the size of the input: the absolute value, respectively norm, larger or smaller than $L_p(\frac{1}{2}, c_M)$, respectively $L_p(\frac{1}{2}, \overline{c_M})$. Finally the descent brings some more parameters: e_i, $\bar{e_i}$, ν_i and $\bar{\nu_i}$ for $i = 1, 2$. The parameters e_i and $\overline{e_i}$ decide the size of the sieving domain in special Q while ν_i and $\overline{\nu_i}$ describe how much we want to descend in one step. For intuition, notice that $i = 1$ suggests a parameter for the rational side while $i = 2$ for the algebraic one. Also an over-lined parameter has the same task as a simple parameter, but it refers to the descent of an ideal rather than a number.

INPUT: a number Q or an ideal $\mathfrak{u}$ of size(norm) less than $L_p(\frac{2}{3}, c)$, q prime factor of $p - 1$;

OUTPUT: $\log_T Q$ mod q.

Algorithm 3 *Descent by special Q*

1. ***if*** *the input is a number* ***then*** *run procedure NUMBER* ***else*** *run procedure IDEAL* ***endif***.
2. *procedure NUMBER(l):*
   (a) ***if*** $|l| < L_p(\frac{1}{2}, c_M)$ ***then*** $\nu = \nu_1$ ***else*** $\nu = \nu_2$;***endif***.
   (b) *search a pair* (a, b) *such that:*
      (i) $|a|, |b| \leq L_p(\frac{1}{3}, e) \cdot |l|^{\frac{1}{2}}$ *and* $gcd(a, b) = 1$;
      (ii) $l \mid a - bm$ *and* $\frac{a-bm}{l}$ *is* l^ν*-smooth;*
      (iii) *for some* $1 \leq j \leq V$ $(a - b\alpha_j)$ *is* $|l|^\nu$*-smooth;*
   (c) *apply procedure NUMBER to all prime factors of* $a - bm$ *and procedure IDEAL to all prime ideal factors of* $(a - b\alpha_j)$;
   (d) *regroup the relations of the factors to obtain a relation for* l.
3. *procedure IDEAL(*$\mathfrak{u}$*):*
   (a) $n = N(\mathfrak{u})$; ***if*** $n < L_p(\frac{1}{2}, \bar{c_M})$ ***then*** $\bar{\nu} = \bar{\nu_1}$ ***else*** $\bar{\nu} = \bar{\nu_2}$;***endif***
   (b) *search a pair* (a, b) *such that*
      (i) $|a|, |b| \leq L_p(\frac{1}{3}, \bar{e}) \cdot n^{\frac{1}{2}}$ *and* $gcd(a, b) = 1$;
      (ii) $(a - b\alpha_j) \cdot \mathfrak{u}^{-1}$ *is* $n^{\bar{\nu}}$*-smooth, where* j *is such that* $\mathfrak{u} \subset K_j$;

(iii) $a - b\alpha_j \in u$ *and* $(a - b\alpha_j)$ *is* $n^{\bar{\nu}}$*-smooth;*

(c) apply algorithm 3 to all prime factors of $a - bm$ *and* $a - b\alpha_j$*.*

(d) regroup the relations of the factors to obtain a relation for $\mathfrak{u}$*.*

Complexity Analysis Since the descent algorithm is a divide and conquer one, the total time is:

$$(\text{number of calls of } \textit{Algorithm } 3) \cdot \textit{time}(\textit{Algorithm } 3).$$

We will show that the number of calls of *Algorithm* 3 is $e^{\mathcal{O}((\log\log p)^2)}$ and $\textit{time}(\textit{Algorithm } 3)$ is $L_p(\frac{1}{3}, 1.188)^{1+o(1)}$ for Discrete logarithm Factory and $L_p(\frac{1}{3}, 1.134)^{1+o(1)}$ for the algorithm of Commeine and Semaev. Therefore

$$\textit{time}(\text{Descent Step}) = L_p(\frac{1}{3}, 1.188)^{1+o(1)}.$$

Number of Applications of Algorithm 3 We reduce a number of size $L(\frac{2}{3})$ into numbers and ideals of size $L(\frac{1}{3})$. We are computing the number of knots of a tree of height w and where each knot has at most Z sons. Since Z is the number of (rational and algebraic) divisors of a number or ideal of norm less than p and since all numbers and ideals have the norm at least 2, we have $Z \leq 2 \cdot \log_2 p = \mathcal{O}(e^{\log\log p})$. For w one picks the smallest value such that: $L_p(\frac{1}{3})^{\nu_i^w} \geq L_p(\frac{2}{3})$ and $L_p(\frac{1}{3})^{\bar{\nu_i}^w} \geq L_p(\frac{2}{3})$ for $i = 1, 2$. Hence we have: $w = \frac{\log(\log L_p(\frac{2}{3})/\log L_p(\frac{1}{3}))}{min(\nu_1,\nu_2,\bar{\nu_1},\bar{\nu_2})} = \mathcal{O}(\log(\log^{\frac{1}{3}} p)) = \mathcal{O}(\log\log p)$. Therefore, the number of calls is: $\frac{Z^w - 1}{Z-1} < Z^w = \mathcal{O}(e^{\log\log p})^{\mathcal{O}(\log\log p)} = e^{\mathcal{O}((\log\log p)^2)}$.□

Time(Algorithm 3)

case ν_1 $|l| < L(\frac{1}{2}, c_M)$

Let's call $P_{1Q} = Prob((a - bm) \text{ is } |l|^{\nu_2} - smooth)$. By the computations below $P_{1Q} = L(\frac{1}{3}, -\frac{1}{3\delta\beta\nu_1})$. Let's put $P_{1A} = Prob((a - b\alpha) \text{ is } |l|^{\nu_2})$. We show below that $P_{1A} = L(\frac{1}{3}, -(\frac{\delta}{6\nu_1} + \frac{1}{3\delta\nu_1\beta} + \frac{e_1\delta}{3\nu_1\beta}))$. The total sieve space has approximately $\frac{(L(\frac{1}{3}, e_1)|l|^{\frac{1}{2}})^2}{|l|}$ elements because we are only using the pairs (a, b) in a square of $\mathbb{Z} \times \mathbb{Z}$ which belong to a lattice of volume $|l|$. Thus the sieve space is $L(\frac{1}{3}, 2e_1)$. The search of (a, b) succeeds if $L(\frac{1}{3}, 2e_1) \cdot P_{1Q} \cdot (VP_{1A}) \geq 1$ where V is the number of fields we use. This comes to $2e_1 \geq (\frac{1}{3\delta\beta\nu_1}) + [(\frac{\delta}{6\nu_1} + \frac{1}{3\delta\nu_1\beta} + \frac{e_1\delta}{3\nu_1\beta}) - (\beta - \gamma)]$. We fulfill this condition by taking $e_1 = (\frac{3\nu_1\beta}{6\nu_1\beta - \delta})(\frac{2}{3\nu_1\delta\beta} + \frac{\delta}{6\nu_1} - \beta + \gamma)$. We do the search by testing with ECM the smoothness of a number, then for every good pair (a, b) we search by

ECM a j such that $(a - b\alpha_j)$ is smooth. There are $L(\frac{1}{3}, 2e_1)$ tests over $\mathbb{Q}$ and $\frac{1}{VP_{1A}}$ tests over the fields. The time of a ECM test is the same for both rational and algebraic side: $L_{(|l|^{\nu_1})}(\frac{1}{2}, \sqrt{2}) = L(\frac{1}{4})$. Therefore, the time-cost of this case is: $t_1 = L(\frac{1}{4}) \cdot (L(\frac{1}{3}, 2e_1) + V^{-1}P_{1A}^{-1}) = L(\frac{1}{3}, max(2e_1, ((\frac{\delta}{6\nu_1} + \frac{1}{3\delta\nu_1\beta} + \frac{e_1\delta}{3\nu_1\beta}) - (\beta - \gamma)))$. This is $L(\frac{1}{3}, 2e_1))$ by the choice of e_1. Now $\nu_1 \mapsto e_1(\nu_1)$ is a function onto $\mathbb{R}_+$ using the formula $\nu_1 = \frac{4+\beta\delta^2+(2e_1)\delta^2}{6\delta\beta(\beta-\gamma+(2e_1))}$. The only problem is that ν_1 might be outside $(0, 1)$. We try to put $e_1 = \frac{c_r}{2}$ where the time for case ν_2 is $t_2 = L(\frac{1}{3}, c_r)$. If $\nu_1 \geq 1$ we increase e_1 gradually.

case ν_2 $|l| \geq L(\frac{1}{2}, c_M)$
Let's call $P_{2Q} = Prob((a - bm) \text{ is } |l|^{\nu_2} - smooth)$. By the computations below $P_{2Q} = L(\frac{1}{6})$. Let's put $P_{2A} = Prob((a - b\alpha) \text{ is } |l|^{\nu_2})$. We show below that $P_{2A} = L(\frac{1}{3}, -\frac{\delta}{6\nu_2})$. The search of (a, b) succeeds if $L(\frac{1}{3}, 2e_2) \cdot P_{2Q} \cdot (VP_{2A}) \geq 1$ where V is the number of fields we use. It comes to $e_2 \geq \frac{1}{2}(\gamma - \beta + \frac{\delta}{6\nu_2})$. We fulfill this condition by taking $e_2 = \frac{1}{2}(\gamma - \beta + \frac{\delta}{6\nu_2})$. Similarly to the case ν_1 the time-cost is: $t_2 = time(\text{a test } ECM) \cdot (L(\frac{1}{3}, 2e_2) + \frac{1}{VP_{2A}})$. A test ECM takes more time here because the smoothness bound is larger: $L_{|l|^{\nu_2}}(\frac{1}{2}, \sqrt{2}) = L(\frac{1}{3}, 2 \cdot 3^{-\frac{2}{3}}\sqrt{\nu_2})$. Therefore $t_2 = L(\frac{1}{3}, 2\sqrt{\frac{\nu_2 c}{3}}) \cdot [L(\frac{1}{3}, 2e_2) + L(\frac{1}{3}, \frac{\delta}{6\nu_2} - (\beta - \gamma))] = L(\frac{1}{3}, 2\sqrt{\frac{\nu_2 c}{3}}) \cdot 2L(\frac{1}{3}, 2e_2) = = L(\frac{1}{3}, 2 \cdot 3^{-\frac{2}{3}}\sqrt{\nu_2} + \frac{\delta}{6\nu_2} - (\beta - \gamma))$. We minimize t_2 by taking $\nu_2 = (\frac{\delta^2}{12c})^{1/3}$.

case $\bar{\nu}_1$ $N(\mathfrak{u}) < L(\frac{1}{2}, \bar{c}_M)$
Put as above $n = N(\mathfrak{u})$. Like while reducing a number put $\bar{P}_{1Q} = Prob((a - bm) \text{ is } n^{\bar{\nu}_1} - smooth)$ and $\bar{P}_{1A} = Prob((a - b\alpha) \text{ is } n^{\bar{\nu}_1})$. According to 1 we have $\bar{P}_{1Q} = L(\frac{1}{3}, -\frac{1}{3\delta\beta\nu_1})$ and $\bar{P}_{1A} = L(\frac{1}{3}, \frac{\delta}{6\bar{\nu}_1} + \frac{1}{3\delta\bar{\nu}_1\beta} + \frac{e_1\delta}{3\bar{\nu}_1\beta})$. We do the search in the same way. What changes is that we need that $(a - b\alpha_j)$ is smooth exactly in the field K_j which contains $\mathfrak{u}$. This changes the condition into $L(\frac{1}{3}, 2\bar{e}_1) \cdot \bar{P}_{1Q} \cdot (\bar{P}_{1A}) \geq 1$. Hence $2\bar{e}_1 \geq (\frac{1}{3\delta\beta\bar{\nu}_1}) + [(\frac{\delta}{6\bar{\nu}_1} + \frac{1}{3\delta\bar{\nu}_1\beta} + \frac{\bar{e}_1\delta}{3\bar{\nu}_1\beta})]$. As usual we take $\bar{e}_1 = (\frac{3\bar{\nu}_1\beta}{6\bar{\nu}_1\beta - \delta})(\frac{2}{3\bar{\nu}_1\delta\beta} + \frac{\delta}{6\bar{\nu}_1})$. The time-cost of a ECM test here doesn't change with respect to case ν_1 because the smoothness bound is the same. Now we can compute the time-cost: $\bar{t}_1 = L(\frac{1}{3}, 2\bar{e}_1) + \bar{P}_{1A}^{-1} = L(\frac{1}{3}, max(2\bar{e}_1, \frac{1}{3\delta\bar{\nu}_1\beta} + \frac{\delta\bar{e}_1}{\bar{\nu}_1\beta})) = L(\frac{1}{3}, 2\bar{e}_1)$. Just like in the case ν_1 we have $\bar{\nu}_1 = \frac{4+\delta^2\gamma+\bar{e}_1\delta^2}{6\delta\gamma\bar{e}_1}$ with $\bar{e}_1 = \frac{c_r}{2}$ (with c_r as above) or the smallest value which gives a $\bar{\nu}_1$ in $(0, 1)$.

case $\bar{\nu}_2$ $N(\mathfrak{u}) \geq L(\frac{1}{2}, \bar{c}_M)$
Put as above $n = N(\mathfrak{u})$. We use $\bar{P}_{2Q} = L(\frac{1}{6})$ and $\bar{P}_{2A} = L(\frac{1}{3}, -\frac{\delta}{6\bar{\nu}_2})$. The condition, same as in case $\bar{\nu}_1$, is fulfilled with $\bar{e}_2 = \frac{\delta}{6\bar{\nu}_2}$. An ECM test takes time $L_{(n^{\bar{\nu}_2})}(\frac{1}{2}, \sqrt{2})$. Since all the ideals to be reduced come from

the descent of some large l we have $n \leq L(\frac{2}{3}, c)^{\nu_2}$. Therefore ECM takes time $L(\frac{1}{3}, 2\sqrt{\frac{\bar{\nu}_2 \nu_{2c}}{3}})$. Hence we have: $\bar{t}_2 = L(\frac{1}{3}, 2\sqrt{\frac{\nu_2 \bar{\nu}_{2c}}{3}})(L(\frac{1}{6}) + L(\frac{1}{3}, \frac{\delta}{6\bar{\nu}_2}))$ $= L(\frac{1}{3}, \frac{\delta}{6\bar{\nu}_2} + 2\sqrt{\frac{\nu_2 \bar{\nu}_{2c}}{3}})$. We minimize $\bar{t}_2$ for $\bar{\nu}_2 = (\frac{\delta^2}{12\nu_{2c}})^{\frac{1}{3}}$.

Numerical Application Using the information above we can compute the best values for the parameters as well as the complexity of the descents. We obtain time $L(\frac{1}{3}, 1.145)$ if one made steps 1 and 2 with parameters as in NFS, $L(\frac{1}{3}, 1.134)$ for the algorithm of Commeine and Semaev and $L(\frac{1}{3}, 1.188)$ the Discrete Logarithm Factory. Notice that for the Discrete Logarithm Factory the descent is longer than in the normal case because we descent lower, i.e. β is smaller. Despite a low smoothness bound, the descent is fast for Commeine and Semaev's algorithm because the multiple fields help us to find relations during descent. The same computations allow us to find the complexity of the descent step for arbitrary values of β.

Computing the probabilities

P_{1Q} We have: $\frac{|a+bm|}{l} = \frac{L(\frac{1}{3}, e_1)|l|^{\frac{1}{2}} L(\frac{2}{3}, \frac{1}{\delta})}{|l|} = L(\frac{2}{3}, \frac{1}{\delta})|l|^{-\frac{1}{2}}$. Further we have: $P_{1Q} = P_{smooth}(L(\frac{2}{3}, \frac{1}{\delta})|l|^{-\frac{1}{2}}, |l|^{\nu_1}) = u^{-u}$ where $u = \frac{\log(L(\frac{2}{3}, \frac{1}{\delta})|l|^{-\frac{1}{2}})}{\nu_1 \log |l|} = -\frac{1}{2\nu_1} + \frac{\log L(\frac{2}{3}, \frac{1}{\delta})}{\nu_1 \log l}$. Since the map $u \mapsto u^{-u}$ is decreasing, the smallest, i. e. worst, probability occurs when $u = u_{max}$. At its turn $l \mapsto u(l)$ is decreasing, so $u_{max} = u(l_{min}) = u(L(\frac{1}{3}, \beta))$. Thus $u_{max} = (1+o(1))\frac{1}{\delta \nu_1 \beta}(\log p)^{\frac{1}{3}}(\log \log p)^{-\frac{1}{3}}$. Therefore $P_{1Q} \geq (-\frac{1}{2\nu_1} + \frac{\log L(\frac{2}{3}, \frac{1}{\delta})}{\nu_1 \log L(\frac{1}{3}, \beta)})^{-\frac{1}{3}} \log \log p = L(\frac{1}{3}, -\frac{1}{3\delta\beta\nu_1})$.

P_{1A} We have for all j: $N(a - b\alpha_j) \leq dL(\frac{2}{3}, \frac{1}{\delta})(L(\frac{1}{3}, e_1)|l|^{\frac{1}{2}})^d = L(\frac{2}{3}, \frac{1}{\delta} + e_1\delta)|l|^{\frac{d}{2}}$. Hence $\log N(a - b\alpha_j) = \log |l| \frac{\delta}{2}(\log p)^{\frac{1}{3}}(\log \log p)^{-\frac{1}{3}} + (\frac{1}{\delta} + e_1\delta)(\log p)^{\frac{2}{3}}(\log \log p)^{\frac{1}{3}}$. We also have $\log(|l|^{\nu_1}) = \nu_1 \log |l|$. Therefore $u = \frac{1}{\nu_1}(\frac{\delta}{2}(\log p)^{\frac{1}{3}}(\log \log p)^{-\frac{1}{3}} + \frac{1}{\log |l|}(\frac{1}{\delta} + e_1\delta)(\log p)^{\frac{2}{3}}(\log \log p)^{\frac{1}{3}})$. As before we use the maximum value of u which is $u_{max} = \frac{1}{\nu_1}(\frac{\delta}{2}(\log p)^{\frac{1}{3}}(\log \log p)^{-\frac{1}{3}} + \frac{1}{\log L(\frac{1}{3}, \beta)}(\frac{1}{\delta} + e_1\delta)(\log p)^{\frac{2}{3}}(\log \log p)^{\frac{1}{3}})$ $= L(\frac{1}{3}, \frac{\delta}{2\nu_1} + \frac{1}{\delta\nu_1\beta} + \frac{e_1\delta}{\nu_1\beta})$. Hence $\log u_{max} = (1 + o(1))\frac{1}{3} \log p$. Which leads to $P_{1A} \geq L(\frac{1}{3}, -(\frac{\delta}{6\nu_1} + \frac{1}{3\delta\nu_1\beta} + \frac{e_1\delta}{3\nu_1\beta}))$.

P_{2Q} It is the probability that a number of size $L(\frac{2}{3})$ is $L(\frac{1}{2})$ − *smooth* because $n \geq L(\frac{1}{2})$. It is $L(\frac{1}{6})^{-1}$.

P_{2A} The computations are the same as in case P_{1A} until we compute u.Then $u_{max} = \frac{1}{\nu_2}(\frac{\delta}{2}(\log p)^{\frac{1}{3}}(\log \log p)^{-\frac{1}{3}} + \frac{1}{\log L(\frac{1}{2}, c_M)}(\frac{1}{\delta} + e_1\delta)\log^{\frac{2}{3}} p(\log \log p)^{\frac{1}{3}})$ be-

cause l is larger. Thus $u_{max} = (1 + o(1))\frac{\delta}{2\nu_2}(\log p)^{\frac{1}{3}} \log\log^{-\frac{1}{3}} p$. Hence $\log u_{max} = (1 + o(1))\frac{1}{3}\log p$ and $P_{2A} \geq L(\frac{1}{3}, -\frac{\delta}{6\nu_2})$.

$\bar{P}_{1Q}$ The difference with P_{1Q} is that we have to replace ν by $\bar{\nu}$ and e_1 by $\bar{e}_1$, then to add $\frac{1}{\bar{\nu}_1}$ to u. Indeed we used to test for smoothness the number $\frac{a-bm}{l}$ while here we test $a - b\alpha_j$. Since $u \to \infty$, adding a constant doesn't change the result in a first approximation. Therefore $\bar{P}_{1Q} = L(\frac{1}{3}, -\frac{1}{3\delta\beta\bar{\nu}_1})$.

$\bar{P}_{1A}$ Instead of testing for smoothness $(a - b\alpha_j)$ as in P_{1A}, we test $(a - b\alpha_j) \cdot \mathfrak{u}^{-1}$. This changes the expected value of u by subtracting $\frac{1}{\bar{\nu}_1}$. It doesn't change the result, so $\bar{P}_{1A} = L(\frac{1}{3}, \frac{\delta}{6\bar{\nu}_1} + \frac{1}{3\delta\bar{\nu}_1\beta} + \frac{e_1\delta}{3\bar{\nu}_1\beta})$.

$\bar{P}_{2Q}$ Just like in P_{2Q}, we have $\bar{P}_{2Q} = L(\frac{1}{6})^{-1}$.

$\bar{P}_{2A}$ The difference with the expected value of u by analogy with P_{2A} is that we have to subtract $\mathcal{O}((\log p)^{\frac{1}{6}})$. It doesn't count because u has size $(\log p)^{\frac{1}{3}}$. Thus $\bar{P}_{2A} = L(\frac{1}{3}, -\frac{\delta}{6\bar{\nu}_2})$.

Chapter 4

Conclusion and Perspectives

The present report pushes further a series of algorithms evolved from Index Calculus. On the one hand, we showed that the idea of Factorization Factory is not incompatible with the discrete logarithm problem. Thus we described a DLP algorithm called Discrete Logarithm Factory with complexity $L_p(\frac{1}{3}, 1.639)$. On the other hand we speeded up the fastest algorithm in practice by adapting the idea of early abort to the NFS family. It brings the complexity of individual logarithm to $L_p(\frac{1}{3}, 1.232)$ which can have consequences for DLP-based cryptosystems.

There are questions which remain open and further improvements which might happen in the field. Let us make a non exhaustive list:

1. Pushing the early abort strategy to its limits. Indeed, in our analysis we showed that 8 admissibility tests bring the complexity to $L_p(\frac{1}{3}, 1.232)^{1+o(1)}$. Can one adapt the number of tests to the size of p in order to speed up the theoretical complexity?

2. Adapting the techniques of descent and early abort to other algorithms. Indeed the similarities between $\mathbb{F}_p$ and $\mathbb{F}_{p^n}$ might lead to similar algorithms.

3. Improving the polynomial selection. Indeed, the difference of complexity between NFS and the Special Number Field Sieve (SNFS) is a consequence of the fact that in the SNFS the coefficients of the polynomial are in $\mathcal{O}(1)$. Despite a negative result in [BLP93] which showed that there are values of p where no improvements can be made, we ignore the existence of a better choice of the polynomials which could speed up NFS on average or for a non negligible fraction of inputs.

Bibliography

[Adl08] L. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 55–60. IEEE, 2008.

[Bac84] E. Bach. Discrete logarithms and factoring. Technical Report CSD–84–186, University of California at Berkley, 1984.

[BLP93] J.P Buhler, H.W. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. In A.K Lenstra and H.W. lenstra (eds), editors, *The development of the Number Fiel Sieve*, volume 1554 of *Lecture Notes in Mathematics*, pages 50–94. Springer–Verlang, 1993.

[Bou70] V. Bouniakowsky. –. *Bulletin of the Academy of Science in Sankt Petersburg*, 1870.

[CEP83] E. Canfield, P. Erdös, and C. Pomerance. On a problem of oppenheim concerning "factoraisatio numerorum". *Journal of Number Theory*, 17:1–28, 1983.

[CL84] H. Cohen and H. Lenstra. Heuristics on class groups of number fields. *Number Theory Noordwijkerhout 1983*, pages 33–62, 1984.

[Coh93] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer–Verlang, 1993.

[Cop93] D. Coppersmith. Modifications to the number field sieve. *J. of Cryptology*, 6:169–180, 1993.

[Cop02] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *Information Theory, IEEE Transactions on*, 30(4):587–594, 2002.

[COS86] D. Coppersmith, AM Odlyzko, and R. Schroeppel. Discrete logarithms in GF (p), Algorithmica. *Vol*, 1:1–15, 1986.

[CS06] An Commeine and Igor Semaev. An algorithm to solve the discrete logarithm problem with the number field sieve. In *Public Key Cryptography*, pages 174–190, 2006.

[CW08] D. Coppersmith and S. Winograd. On the asymptotic complexity of matrix multiplication. In *Foundations of Computer Science, 1981. SFCS'81. 22nd Annual Symposium on*, pages 82–90. IEEE, 2008.

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

[ElG02] T. ElGamal. A subexponential-time algorithm for computing discrete logarithms over $GF(p^2)$. *Information Theory, IEEE Transactions on*, 31(4):473–481, 2002.

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

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

[Kra22] M. Kraïtchik. *Théorie des nombres*. Gauthier–Villars, 1922.

[Len87] H.W. Lenstra. Factoring integers with elliptic curves. *Annals of mathematics*, 126(3):649–673, 1987.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT–24:106–110, 1978.

[Poh77] S. C. Pohling. Algebraic and combinatoric aspects of cryptography. Technical report, Stanford University, October 1977.

[Pol93] J. Pollard. The lattice sieve. *The development of the number field sieve*, pages 43–49, 1993.

[Pom82] C. Pomerance. Analysis and comparison of some integer factoring algorithms. *Mathematisch Centrum Computational Methods in Number Theory, Pt. 1 p 89-139(SEE N 84-17990 08-67)*, 1982.

[Pom08] C. Pomerance. A tale of two sieves. *Biscuits of Number Theory*, page 85, 2008.

[Sch93] O. Schirokauer. Discrete logarithms and local units. *Philosophical Transactions of the Royal Society London, series A*, 345:409–424, 1993.

[Sch05] O. Schirokauer. Virtual logarithms. *Journal of Algorithms*, 57:140–147, 2005.

[Sho90] V. Shoup. Searching for primitive roots in finite fields. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 546–554. ACM, 1990.

[Sho09] V. Shoup. *A computational introduction to number theory and algebra*. Cambridge Univ Pr, 2009.

[Sil87] R.D. Silverman. The multiple polynomial quadratic sieve. *Mathematics of Computation*, 48(177):329–339, 1987.

[Sta07] HM Stark. The Gauss Class-Number Problems. *Analytic number theory: a tribute to Gauss and Dirichlet*, page 247, 2007.

[Wie86] D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inform. Theory*, IT–32(1):54–62, 1986.

Appendix

Here is a diagram representing algorithms of the Number Field Sieve family. Each box contains the name of an algorithm followed by its complexity, its author and year of publication. The color code is red for factorization and blue for discrete logarithm. For DLP algorithms, when we write "x, then y", x denotes the time of the precomputations while y is the time of the individual logarithm. If the algorithm was presented earlier in a conference or announced by personal communication we include the year in parenthesis.