Chapter 15

Factoring and Discrete Logarithms in Subexponential Time

One of the most powerful tools in mathematics is linear algebra, and much of mathematics is devoted to solving problems by reducing them to it. It is therefore natural to try to solve the integer factorisation and discrete logarithm problems (DLP) in this way. This chapter briefly describes a class of algorithms that exploit a notion called "smoothness", to reduce factoring or DLP to linear algebra. We present such algorithms for integer factorisation, the DLP in the multiplicative group of a finite field, and the DLP in the divisor class group of a curve.

It is beyond the scope of this book to give all the details of these algorithms. Instead, the aim is to sketch the basic ideas. We mainly present algorithms with nice theoretical properties (though often still requiring heuristic assumptions) rather than the algorithms with the best practical performance. We refer to Crandall and Pomerance [162], Shoup [556] and Joux [317] for further reading.

The chapter is arranged as follows. First we present results on smooth integers, and then sketch Dixon's random squares factoring algorithm. Section 15.2.3 then summarises the important features of all algorithms of this type. We then briefly describe a number of algorithms for the discrete logarithm problem in various groups.

15.1 Smooth Integers

Recall from Definition 12.3.1 that an integer is B-smooth if all its prime divisors are at most B. We briefly recall some results on smooth integers; see Granville [267] for a survey of this subject and for further references.

Definition 15.1.1. Let $X, Y \in \mathbb{N}$ be such that $2 \leq Y < X$. Define

$$\Psi(X,Y) = \#\{n \in \mathbb{N} : 1 \leq n \leq X,\ n \text{ is } Y\text{-smooth}\}.$$

It is important for this chapter to have good bounds on $\Psi(X,Y)$. Let $u = \log(X)/\log(Y)$ (as usual log denotes the natural logarithm), so that $u > 1$, $Y = X^{1/u}$ and $X = Y^u$. There is a function $\rho : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ called the **Dickman-de Bruijn function** (for the exact definition of this function see Section 1.4.5 of [162]) such that, for fixed $u > 1$, $\Psi(X, X^{1/u}) \sim X\rho(u)$, where $f(X) \sim g(X)$ means $\lim_{X\to\infty} f(X)/g(X) = 1$. A crude estimate for $\rho(u)$, as $u \to \infty$ is $\rho(u) \approx 1/u^u$. For further details and references see Section 1.4.5 of [162].

The following result of Canfield, Erdös and Pomerance [117] is the main tool in this subject. This is a consequence of Theorem 3.1 (and the corollary on page 15) of [117].

Theorem 15.1.2. *Let $N \in \mathbb{N}$. Let $\epsilon, u \in \mathbb{R}$ be such that $\epsilon > 0$ and $3 \leq u \leq (1-\epsilon)\log(N)/\log(\log(N))$. Then there is a constant c_ϵ (that does not depend on u) such that*

$$\Psi(N, N^{1/u}) = N\exp(-u(\log(u) + \log(\log(u)) - 1 + (\log(\log(u)) - 1)/\log(u) + E(N,u))) \tag{15.1}$$

where $|E(N,u)| \leq c_\epsilon(\log(\log(u))/\log(u))^2$.

Corollary 15.1.3. *Let the notation be as in Theorem 15.1.2. Then $\Psi(N, N^{1/u}) = Nu^{-u+o(u)} = Nu^{-u(1+o(1))}$ uniformly as $u \to \infty$ and $u \leq (1-\epsilon)\log(N)/\log(\log(N))$ (and hence also $N \to \infty$).*

Exercise 15.1.4. Prove Corollary 15.1.3.
[Hint: Show that the expression inside the exp in equation (15.1) is of the form $-u\log(u) + o(u)\log(u)$.]

We will use the following notation throughout the book.

Definition 15.1.5. Let $0 \leq a \leq 1$ and $c \in \mathbb{R}_{>0}$. The **subexponential function** for the parameters a and c is

$$L_N(a,c) = \exp(c\log(N)^a \log(\log(N))^{1-a}).$$

Note that taking $a = 0$ gives $L_N(0,c) = \log(N)^c$ (polynomial) while taking $a = 1$ gives $L_N(1,c) = N^c$ (exponential). Hence $L_N(a,c)$ interpolates exponential and polynomial growth. A complexity $O(L_N(a,c))$ with $0 < a < 1$ is called **subexponential**.

Lemma 15.1.6. *Let $0 < a < 1$ and $0 < c$.*

1. *$L_N(a,c)^m = L_N(a, mc)$ for $m \in \mathbb{R}_{>0}$.*
2. *Let $0 < a_1, a_2 < 1$ and $0 < c_1, c_2$. Then, where the term $o(1)$ is as $N \to \infty$,*

$$L_N(a_1,c_1) \cdot L_N(a_2,c_2) = \begin{cases} L_N(a_1, c_1 + o(1)) & \text{if } a_1 > a_2, \\ L_N(a_1, c_1 + c_2) & \text{if } a_1 = a_2, \\ L_N(a_2, c_2 + o(1)) & \text{if } a_2 > a_1. \end{cases}$$

3.

$$L_N(a_1,c_1) + L_N(a_2,c_2) = \begin{cases} O(L_N(a_1,c_1)) & \text{if } a_1 > a_2, \\ O(L_N(a_1, \max\{c_1,c_2\} + o(1))) & \text{if } a_1 = a_2, \\ O(L_N(a_2,c_2)) & \text{if } a_2 > a_1. \end{cases}$$

4. *Let $0 < b < 1$ and $0 < d$. If $M = L_N(a, c)$ then $L_M(b, d) = L_N(ab, dc^b a^{1-b} + o(1))$ as $N \to \infty$.*
5. *$\log(N)^m = O(L_N(a, c))$ for any $m \in \mathbb{N}$.*
6. *$L_N(a, c)\log(N)^m = O(L_N(a, c + o(1)))$ as $N \to \infty$ for any $m \in \mathbb{N}$. Hence, one can always replace $\tilde{O}(L_N(a, c))$ by $O(L_N(a, c + o(1)))$.*
7. *$\log(N)^m \leq L_N(a, o(1))$ as $N \to \infty$ for any $m \in \mathbb{N}$.*
8. *If $F(N) = O(L_N(a, c))$ then $F(N) = L_N(a, c + o(1))$ as $N \to \infty$.*
9. *$L_N(1/2, c) = N^{c\sqrt{\log(\log(N))/\log(N)}}$.*

Exercise 15.1.7. Prove Lemma 15.1.6.

Corollary 15.1.8. *Let $c > 0$. As $N \to \infty$, the probability that a randomly chosen integer $1 \leq x \leq N$ is $L_N(1/2, c)$-smooth is $L_N(1/2, -1/(2c) + o(1))$.*

Exercise 15.1.9. Prove Corollary 15.1.8 (using Corollary 15.1.3).

Exercise 15.1.10. Let $0 < b < a < 1$. Let $1 \leq x \leq L_N(a, c)$ be a randomly chosen integer. Show that the probability that x is $L_N(b, d)$-smooth is $L_N(a-b, -c(a-b)/d+o(1))$ as $N \to \infty$.

15.2 Factoring using Random Squares

The goal of this section is to present a simple version of Dixon's random squares factoring algorithm. This algorithm is easy to describe and analyse, and already displays many of the important features of the algorithms in this chapter. Note that the algorithm is not used in practice. We give a complexity analysis and sketch how subexponential running times naturally arise. Further details about this algorithm can be found in Section 16.3 of Shoup [556] and Section 19.5 of von zur Gathen and Gerhard [238].

Let $N \in \mathbb{N}$ be an integer to be factored. We assume in this section that N is odd, composite and not a perfect power. As in Chapter 12 we focus on splitting N into a product of two smaller numbers (neither of which is necessarily prime). The key idea is that if one can find congruent squares

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

such that $x \not\equiv \pm y \pmod{N}$ then one can split N by computing $\gcd(x - y, N)$.

Exercise 15.2.1. Let N be an odd composite integer and m be the number of distinct primes dividing N. Show that the equation $x^2 \equiv 1 \pmod{N}$ has 2^m solutions modulo N.

A general way to find congruent squares is the following.[1] Select a **factor base** $\mathcal{B} = \{p_1, \ldots, p_s\}$ consisting of the primes $\leq B$ for some $B \in \mathbb{N}$. Choose uniformly at random an integer $1 \leq x < N$, compute $a = x^2 \pmod{N}$ reduced to the range $1 \leq a < N$ and try to factor a as a product in $\mathcal{B}$ (e.g., using trial division).[2] If a is B-smooth then this succeeds, in which case we have a **relation**

$$x^2 \equiv \prod_{i=1}^{s} p_i^{e_i} \pmod{N}. \tag{15.2}$$

[1] This idea goes back to Kraitchik in the 1920s; see [489] for some history.

[2] To obtain non-trivial relations one should restrict to integers in the range $\sqrt{N} < x < N - \sqrt{N}$. But it turns out to be simpler to analyse the algorithm for the case $1 \leq x < N$. Note that the probability that a randomly chosen integer $1 \leq x < N$ satisfies $1 \leq x < \sqrt{N}$ is negligible.

The values x for which a relation is found are stored as $x_1, x_2, \ldots, x_t$. The corresponding exponent vectors $\underline{e}_j = (e_{j,1}, \ldots, e_{j,s})$ for $1 \leq j \leq t$ are also stored. When enough relations have been found we can use linear algebra modulo 2 to obtain congruent squares. More precisely, compute $\lambda_j \in \{0, 1\}$ such that not all $\lambda_j = 0$ and

$$\sum_{j=1}^{t} \lambda_j \underline{e}_j \equiv (0, 0, \ldots, 0) \pmod 2.$$

Equivalently, this is an integer linear combination

$$\sum_{j=1}^{t} \lambda_j \underline{e}_j = (2f_1, \ldots, 2f_s) \tag{15.3}$$

with not all the f_i equal to zero. Let

$$X \equiv \prod_{j=1}^{t} x_j^{\lambda_j} \pmod N, \qquad Y \equiv \prod_{i=1}^{s} p_i^{f_i} \pmod N. \tag{15.4}$$

One then has $X^2 \equiv Y^2 \pmod N$ and one can hope to split N by computing $\gcd(X - Y, N)$ (note that this gcd could be 1 or N, in which case the algorithm has failed). We present the above method as Algorithm 22.

Algorithm 22 Random squares factoring algorithm

INPUT: $N \in \mathbb{N}$
OUTPUT: Factor of N

1: Select a suitable $B \in \mathbb{N}$ and construct the **factor base** $\mathcal{B} = \{p_1, \ldots, p_s\}$ consisting of all primes $\leq B$
2: **repeat**
3: Choose an integer $1 \leq x < N$ uniformly at random and compute $a = x^2 \pmod N$ reduced to the range $1 \leq a < N$
4: Try to factor a as a product in $\mathcal{B}$ (e.g., using trial division)
5: **if** a is B-smooth **then**
6: store the value x and the exponent row vector $\underline{e} = (e_1, \ldots, e_s)$ as in equation (15.2) in a matrix
7: **end if**
8: **until** there are $s + 1$ rows in the matrix
9: Perform linear algebra over $\mathbb{F}_2$ to find a non-trivial linear dependence among the vectors $\underline{e}_j$ modulo 2
10: Define X and Y as in equation (15.4)
11: **return** $\gcd(X - Y, N)$

We emphasise that the random squares algorithm has two distinct stages. The first stage is to generate enough relations. The second stage is to perform linear algebra. The first stage can easily be distributed or parallelised, while the second stage is hard to parallelise.

Example 15.2.2. Let $N = 19 \cdot 29 = 551$ and let $\mathcal{B} = \{2, 3, 5\}$. One finds the following congruences (in general 4 relations would be required, but we are lucky in this case)

$$\begin{aligned} 34^2 &\equiv 2 \cdot 3^3 \pmod N \\ 52^2 &\equiv 2^2 \cdot 5^3 \pmod N \\ 55^2 &\equiv 2 \cdot 3^3 \cdot 5 \pmod N. \end{aligned}$$

These relations are stored as the matrix

$$\begin{pmatrix} 1 & 3 & 0 \\ 2 & 0 & 3 \\ 1 & 3 & 1 \end{pmatrix}.$$

The sum of the three rows is the vector

$$(4,\ 6,\ 4)\,.$$

Let

$$X = 264 \equiv 34 \cdot 52 \cdot 55 \pmod{551} \text{ and } Y = 496 \equiv 2^2 \cdot 3^3 \cdot 5^2 \pmod{551}.$$

It follows that

$$X^2 \equiv Y^2 \pmod{N}$$

and $\gcd(X - Y, N) = 29$ splits N.

Exercise 15.2.3. Factor $N = 3869$ using the above method and factor base $\{2, 3, 5, 7\}$.

15.2.1 Complexity of the Random Squares Algorithm

There are a number of issues to deal with when analysing this algorithm. The main problem is to decide how many primes to include in the factor base. The prime number theorem implies that $s = \#\mathcal{B} \approx B/\log(B)$. If we make B larger then the chances of finding a B-smooth number increase, but on the other hand, we need more relations and the linear algebra takes longer. We will determine an optimal value for B later. First we must write down an estimate for the running time of the algorithm, as a function of s. Already this leads to various issues:

- What is the probability that a random value $x^2 \pmod{N}$ factors over the factor base $\mathcal{B}$?
- How many relations do we require until we can be sure there is a non-trivial vector $\underline{e}$?
- What are the chances that computing $\gcd(X - Y, N)$ splits N?

We deal with the latter two points first. It is immediate that $s+1$ relations are sufficient for line 9 of Algorithm 22 to succeed. The question is whether $1 < \gcd(X - Y, N) < N$ for the corresponding integers X and Y. There are several ways the algorithm can fail to split N. For example, it is possible that a relation in equation (15.2) is such that all e_i are even and $x \equiv \pm \prod_i p_i^{e_i/2} \pmod{N}$. One way that such relations could arise is from $1 \leq x < \sqrt{N}$ or $N - \sqrt{N} < x < N$; this situation occurs with negligible probability. If $\sqrt{N} < x < N - \sqrt{N}$ and $a = Y^2$ is a square in $\mathbb{N}$ then $1 \leq Y < \sqrt{N}$ and so $x \not\equiv \pm Y \pmod{N}$ and the relation is useful. The following result shows that all these (and other) bad cases occur with probability at most $1/2$.

Lemma 15.2.4. *The probability to split N using X and Y is at least $\frac{1}{2}$.*

Proof: Let X and Y be the integers computed in line 10 of Algorithm 22. We treat Y as fixed, and consider the probability distribution for X. By Exercise 15.2.1, the number of solutions Z to $Z^2 \equiv Y^2 \pmod{N}$ is 2^m where $m \geq 2$ is the number of distinct primes dividing N. The two solutions $Z = \pm Y$ are useless but the other $2^m - 2$ solutions will all split N.

Since the values for x are chosen uniformly at random it follows that X is a randomly chosen solution to the equation $X^2 \equiv Y^2 \pmod{N}$. It follows that the probability to split N is $(2^m - 2)/2^m \geq 1/2$. □

Exercise 15.2.5. Show that if one takes $s + l$ relations where $l \geq 2$ then the probability of splitting N is at least $1 - 1/2^l$.

We now consider the probability of smoothness. We first assume the probability that $x^2 \pmod N$ is smooth is the same as the probability that a random integer modulo N is smooth.[3]

Lemma 15.2.6. *Let the notation be as above. Let T_B be the expected number of trials until a randomly chosen integer modulo N is B-smooth. Assuming that squares modulo N are as likely to be smooth as random integers of the same size, Algorithm 22 has expected running time at most*

$$c_1 \#\mathcal{B}^2 T_B M(\log(N)) + c_2 (\#\mathcal{B})^3$$

bit operations for some constants c_1, c_2 (where $M(n)$ is the cost of multiplying two n-bit integers).

Proof: Suppose we compute the factorisation of $x^2 \pmod N$ over $\mathcal{B}$ by trial division. This requires $O(\#\mathcal{B} M(\log(N)))$ bit operations for each value of x. We need $(\#\mathcal{B} + 1)$ relations to have a soluble linear algebra problem. As said above, the expected number of trials of x to get a B-smooth value of $x^2 \pmod N$ is T_B. Hence the cost of finding the relations is $O((\#\mathcal{B} + 1) T_B (\#\mathcal{B}) M(\log(N)))$, which gives the first term.

The linear algebra problem can be solved using Gaussian elimination (we are ignoring that the matrix is sparse) over $\mathbb{F}_2$, which takes $O((\#\mathcal{B})^3)$ bit operations. This gives the second term. □

It remains to choose B as a function of N to minimise the running time. By the discussion in Section 15.1, it is natural to approximate T_B by u^u where $u = \log(N)/\log(B)$. We now explain how subexponential functions naturally arise in such algorithms. Since increasing B makes the linear algebra slower, but makes relations more likely (i.e., lowers T_B), a natural approach to selecting B is to try to equate both terms of the running time in Lemma 15.2.6. This leads to $u^u = \#\mathcal{B}$. Putting $u = \log(N)/\log(B)$, $\#\mathcal{B} = B/\log(B)$, taking logs, and ignoring $\log(\log(B))$ terms, gives

$$\log(N)\log(\log(N))/\log(B) \approx \log(B).$$

This implies $\log(B)^2 \approx \log(N)\log(\log(N))$ and so $B \approx L_N(1/2, 1)$. The overall complexity for this choice of B would be $L_N(1/2, 3 + o(1))$ bit operations.

A more careful argument is to set $B = L_N(1/2, c)$ and use Corollary 15.1.3. It follows that $T_B = L_N(1/2, 1/(2c) + o(1))$ as $N \to \infty$. Putting this into the equation of Lemma 15.2.6 gives complexity $L_N(1/2, 2c + 1/(2c) + o(1)) + L_N(1/2, 3c)$ bit operations. The function $x + 1/x$ is minimised at $x = 1$, hence we should take $c = 1/2$.

Theorem 15.2.7. *Let the notation be as above. Under the same assumptions as Lemma 15.2.6 then Algorithm 22 has complexity*

$$L_N(1/2, 2 + o(1))$$

bit operations as $N \to \infty$.

[3] Section 16.3 of Shoup [556] gives a modification of the random squares algorithm for which one can avoid this assumption. The trick is to note that at least one of the cosets of $(\mathbb{Z}/N\mathbb{Z})^*/((\mathbb{Z}/N\mathbb{Z})^*)^2$ has at least as great a proportion of smooth numbers as random integers up to N (Shoup credits Rackoff for this trick). The idea is to work in one of these cosets by choosing at random some $1 < \delta < N$ and considering relations coming from smooth values of $\delta x^2 \pmod N$.

Proof: Put $B = L_N(1/2, 1/2)$ into Lemma 15.2.6. □

We remark that, unlike the Pollard rho or Pollard $p-1$ methods, this factoring algorithm has essentially no dependence on the factors of N. In other words, its running time is essentially the same for all integers of a given size. This makes it particularly suitable for factoring $N = pq$ where p and q are primes of the same size.

15.2.2 The Quadratic Sieve

To improve the result of the previous section it is necessary to reduce the cost of the linear algebra and to reduce the cost of decomposing smooth elements as products of primes. We sketch the **quadratic sieve** algorithm of Pomerance. We do not have space to present all the details of this algorithm (interested readers should see Section 6.1 of [162] or Section 16.4.2 of [556]).

A crucial idea, which seems to have first appeared in the work of Schroeppel[4], is **sieving**. The point is to consider a range of values of x and simultaneously determine the decompositions of $x^2 \pmod N$ over the factor base. It is possible to do this so that the cost of each individual decomposition is only $O(\log(B))$ bit operations.

Another crucial observation is that the relation matrix is sparse, in other words, rows of the matrix have rather few non-zero entries. In such a case, the cost of linear algebra can be reduced from $O((\#\mathcal{B})^3)$ bit operations to $O((\#\mathcal{B})^{2+o(1)})$ bit operations (as $\#\mathcal{B} \to \infty$). The best methods are due to Lanczos or Wiedemann; see Section 6.1.3 of Crandall and Pomerance [162] or Section 3.4 of Joux [317] for references and discussion.

A further trick is to choose $x = \lfloor \sqrt{N} \rfloor + i$ where $i = 0, 1, -1, 2, -2, \ldots$. The idea is that if $x = \sqrt{N} + \epsilon$ then either $x^2 - N$ or $N - x^2$ is a positive integer of size $2\sqrt{N}|\epsilon|$. Since these integers are much smaller than N they have a much better chance of being smooth than the integers $x^2 \pmod N$ in the random squares algorithm. To allow for the case of $\epsilon < 0$ we need to add -1 to our factor base and use the fact that a factorisation $N - x^2 = \prod_{i=1}^{s} p_i^{e_i}$ corresponds to a relation $x^2 \equiv (-1)\prod_{i=1}^{s} p_i^{e_i} \pmod N$.

Since we are now only considering values x of the form $\sqrt{N} + \epsilon$ where $|\epsilon|$ is small it is necessary to assume the probability that $x^2 - N$ or $N - x^2$ (as appropriate) is B-smooth is that same as the probability that a randomly chosen integer of that size is B-smooth. This is a rather strong assumption (though it is supported by numerical evidence) and so the running time estimates of the quadratic sieve are only heuristic.

The heuristic complexity of the quadratic sieve is determined in Exercise 15.2.8. Note that, since we will need to test $L_N(1/2, 1 + o(1))$ values (here $o(1)$ is as $N \to \infty$) for smoothness, we have $|\epsilon| = L_N(1/2, 1 + o(1))$. It follows that the integers being tested for smoothness have size $\sqrt{N}L_N(1/2, 1 + o(1)) = N^{1/2+o(1)}$.

Exercise 15.2.8. ★ Let T_B be the expected number of trials until an integer of size $2\sqrt{N}L_N(1/2, 1)$ is B-smooth. Show that the running time of the quadratic sieve is at most

$$c_1 \#\mathcal{B} T_B \log(B) M(\log(N)) + c_2 \#\mathcal{B}^{2+o(1)}$$

bit operations for some constants c_1, c_2 as $N \to \infty$.

Let $B = L_N(1/2, 1/2)$. Show that the natural heuristic assumption (based on Corollary 15.1.8) is that $T_B = L_N(1/2, 1/2 + o(1))$. Hence, show that the heuristic complexity of the quadratic sieve is $L_N(1/2, 1 + o(1))$ bit operations as $N \to \infty$.

Example 15.2.9. Let $N = 2041$ so that $\lfloor \sqrt{N} \rfloor = 45$.

Let $\mathcal{B} = \{-1, 2, 3, 5\}$. Taking $x = 43, 44, 45, 46$ one finds the following factorisations of $x^2 - N$:

[4] See [371, 489] for some remarks on the history of integer factoring algorithms.

x	$x^2 \pmod N$	$\underline{e}$
43	$-2^6 \cdot 3$	(1, 6, 1, 0)
44	Not 5-smooth	
45	-2^4	(1, 4, 0, 0)
46	$3 \cdot 5^2$	(0, 0, 1, 2)

Taking $\underline{e} = \underline{e}_1 + \underline{e}_2 + \underline{e}_3 = (2, 10, 2, 2)$ gives all coefficients even. Putting everything together, we set $X = 43 \cdot 45 \cdot 46 \equiv 1247 \pmod N$ and $Y = -1 \cdot 2^5 \cdot 3 \cdot 5 \equiv 1561 \pmod N$. One can check that $X^2 \equiv Y^2 \pmod N$ and that $\gcd(X - Y, N) = 157$.

Exercise 15.2.10. Show that in the quadratic sieve one can also use values $x = \lfloor \sqrt{kN} \rfloor + i$ where $k \in \mathbb{N}$ is very small and $i = 0, 1, -1, 2, -2, \ldots$.

Exercise 15.2.11. Show that using sieving and fast linear algebra, but not restricting to values $\pm x^2 \pmod N$ of size $N^{1/2+o(1)}$ gives an algorithm with heuristic expected running time of $L_N(1/2, \sqrt{2} + o(1))$ bit operations as $N \to \infty$.

Exercise 15.2.12. A subexponential algorithm is asymptotically much faster than a $\tilde{O}(N^{1/4})$ algorithm. Verify that if $N = 2^{1024}$ then $N^{1/4} = 2^{256}$ while $L_N(1/2, 2) \approx 2^{197}$ and $L_N(1/2, 1) \approx 2^{98.5}$.

The best proven asymptotic complexity for factoring integers N is $L_N(1/2, 1 + o(1))$ bit operations. This result is due to Pomerance and Lenstra [381].

15.2.3 Summary

We briefly highlight the key ideas in the algorithms of this section. The crucial concept of smooth elements of the group $(\mathbb{Z}/N\mathbb{Z})^*$ arises from considering an integer modulo N as an element of $\mathbb{Z}$. The three essential properties of smooth numbers that were used in the algorithm are:

1. One can efficiently decompose an element of the group as a product of smooth elements, or determine that the element is not smooth.
2. The probability that a random element is smooth is sufficiently high.
3. There is a way to apply linear algebra to the relations obtained from smooth elements to solve the computational problem.

We will see analogues of these properties in the algorithms below.

There are other general techniques that can be applied in most algorithms of this type. For example, the linear algebra problems are usually sparse and so the matrices and algorithms should be customised for this. Another general concept is "large prime variation" which, in a nutshell, is to also store "nearly smooth" relations (i.e., elements that are the product of a smooth element with one or two prime elements that are not too large) and perform some elimination of these "large primes" before doing the main linear algebra stage (this is similar to, but more efficient than, taking a larger factor base). Finally we remark that the first stage of these algorithms (i.e., collecting relations) can always be distributed or parallelised.

15.3 Elliptic Curve Method Revisited

We assume throughout this section that $N \in \mathbb{N}$ is an integer to be factored and that N is odd, composite, and not a perfect power. We denote by p the smallest prime factor of N.

The elliptic curve method (ECM) works well in practice but, as with the Pollard $p-1$ method, its complexity depends on the size of the smallest prime dividing N. It is not a polynomial-time algorithm because, for any constant $c > 0$ and over all N and $p \mid N$, a randomly chosen elliptic curve over $\mathbb{F}_p$ is not likely to have $O(\log(N)^c)$-smooth order. As we have seen, the theorem of Canfield, Erdös and Pomerance [117] says it is more reasonable to hope that integers have a subexponential probability of being subexponentially smooth. Hence, one might hope that the elliptic curve method has subexponential complexity. Indeed, Lenstra [377] makes the following conjecture (which is essentially that the Canfield-Erdös-Pomerance result holds in small intervals).

Conjecture 15.3.1. *(Lenstra [377], page 670) The probability that an integer, chosen uniformly at random in the range $(X - \sqrt{X}, X + \sqrt{X})$, is $L_X(1/2, c)$-smooth is $L_X(1/2, -1/(2c) + o(1))$ as X tends to infinity.*[5]

One can phrase Conjecture 15.3.1 as saying that, if p_s is the probability that a random integer between 1 and X is Y-smooth, then $\Psi(X + 2\sqrt{X}, Y) - \Psi(X, Y) \approx 2\sqrt{X}p_s$. More generally, one would like to know that, for sufficiently large[6] X, Y and Z,

$$\Psi(X + Z, Y) - \Psi(X, Y) \sim Z\Psi(X, Y)/X \tag{15.5}$$

or, in other words, that integers in a short interval at X are about as likely to be Y-smooth as integers in a large interval at X.

We now briefly summarise some results in this area; see Granville [267] for details and references. Harman (improved by Lenstra, Pila and Pomerance [380]) showed, for any fixed $\beta > 1/2$ and $X \geq Y \geq \exp(\log(X)^{2/3+o(1)})$, where the $o(1)$ is as $X \to \infty$, that

$$\Psi(X + X^\beta, Y) - \Psi(X, Y) > 0.$$

Obtaining results for the required value $\beta = 1/2$ seems to be hard and the experts refer to the "$\sqrt{X}$ barrier" for smooth integers in short intervals. It is known that this barrier can be broken most of the time: Hildebrand and Tenenbaum showed that, for any $\epsilon > 0$, equation (15.5) holds when $X \geq Y \geq \exp(\log(X)^{5/6+\epsilon})$ and $Y \exp(\log(X)^{1/6}) \leq Z \leq X$ for all but at most $M/\exp(\log(M)^{1/6-\epsilon})$ integers $1 \leq X \leq M$. As a special case, this result shows that, for almost all primes p, the interval $[p-\sqrt{p}, p+\sqrt{p}]$ contains a Y-smooth integer where $Y = \exp(\log(X)^{5/6+\epsilon})$ (i.e., subexponential smoothness).

Using Conjecture 15.3.1 one obtains the following complexity for the elliptic curve method (we stress that the complexity is in terms of the smallest prime factor p of N, rather than N itself).

Theorem 15.3.2. *(Conjecture 2.10 of [377]) Assume Conjecture 15.3.1. One can find the smallest factor p of an integer N in $L_p(1/2, \sqrt{2} + o(1))M(\log(N))$ bit operations as $p \to \infty$.*

Proof: Guess the size of p and choose $B = L_p(1/2, 1/\sqrt{2})$ (since the size of p is not known one actually runs the algorithm repeatedly for slowly increasing values of B). Then each run of Algorithm 12 requires $O(B \log(B)M(\log(N))) = L_p(1/2, 1/\sqrt{2} + o(1))M(\log(N))$ bit operations. By Conjecture 15.3.1 one needs to repeat the process $L_p(1/2, 1/\sqrt{2}+o(1))$ times. The result follows. □

[5] Lenstra considers the sub-interval $(X-\sqrt{X}, X+\sqrt{X})$ of the Hasse interval $[X+1-2\sqrt{X}, X+1+2\sqrt{X}]$ because the distribution of isomorphism classes of randomly chosen elliptic curves is relatively close to uniform when restricted to those whose group order lies in this sub-interval. In contrast, elliptic curves whose group orders are near the edge of the Hasse interval arise with lower probability.

[6] The notation $\sim$ means taking a limit as $X \to \infty$, so it is necessary that Y and Z grow in a controlled way as X does.

Exercise 15.3.3. Let $N = pq$ where p is prime and $p < \sqrt{N} < 2p$. Show that $L_p(1/2, \sqrt{2} + o(1)) = L_N(1/2, 1 + o(1))$. Hence, in the worst case, the complexity of ECM is the same as the complexity of the quadratic sieve.

For further details on the elliptic curve method we refer to Section 7.4 of [162]. We remark that Lenstra, Pila and Pomerance [380] have considered a variant of the elliptic curve method using divisor class groups of hyperelliptic curves of genus 2. The Hasse-Weil interval for such curves contains an interval of the form $(X, X + X^{3/4})$ and Theorem 1.3 of [380] proves that such intervals contain $L_X(2/3, c_1)$-smooth integers (for some constant c_1) with probability $1/L_X(1/3, 1)$. It follows that there is a rigorous factoring algorithm with complexity $L_p(2/3, c)$ bit operations for some constant c_2. This algorithm is not used in practice, as the elliptic curve method works fine already.

Exercise 15.3.4. Suppose a sequence of values $1 < x < N$ are chosen uniformly at random. Show that one can find such a value that is $L_N(2/3, c)$-smooth, together with its factorisation, in expected $L_N(1/3, c' + o(1))$ bit operations for some constant c'.

Remark 15.3.5. It is tempting to conjecture that the Hasse interval contains a polynomially-smooth integer (indeed, this has been done by Maurer and Wolf [407]; see equation (21.9)). This is not relevant for the elliptic curve factoring method, since such integers would be very rare. Suppose the probability that an integer of size X is Y-smooth is exactly $1/u^u$, where $u = \log(X)/\log(Y)$ (by Theorem 15.1.2, this is reasonable as long as $Y^{1-\epsilon} \geq \log(X)$). It is natural to suppose that the interval $[X - 2\sqrt{X}, X + 2\sqrt{X}]$ is likely to contain a Y-smooth integer if $4\sqrt{X} > u^u$. Let $Y = \log(X)^c$. Taking logs of both sides of the inequality gives the condition

$$\log(4) + \tfrac{1}{2}\log(X) > \frac{\log(X)}{c\log(\log(X))}(\log(\log(X)) - \log(c\log(\log(X)))).$$

It is therefore natural to conclude that when $c \geq 2$ there is a good chance that the Hasse interval of an elliptic curve over $\mathbb{F}_p$ contains a $\log(p)^c$-smooth integer. Proving such a claim seems to be far beyond the reach of current techniques.

15.4 The Number Field Sieve

The most important integer factorisation algorithm for large integers is the **number field sieve** (NFS). A special case of this method was invented by Pollard.[7] The algorithm requires algebraic number theory and a complete discussion of it is beyond the scope of this book. Instead, we just sketch some of the basic ideas. For full details we refer to Lenstra and Lenstra [372], Section 6.2 of Crandall and Pomerance [162], Section 10.5 of Cohen [136] or Stevenhagen [584].

As we have seen from the quadratic sieve, reducing the size of the values being tested for smoothness yields a better algorithm. Indeed, in the quadratic sieve the numbers were reduced from size $O(N)$ to $O(N^{1/2+o(1)})$ and, as shown by Exercise 15.2.11, this trick alone lowers the complexity from $O(L_N(1/2, \sqrt{2} + o(1)))$ to $O(L_N(1/2, 1 + o(1)))$. To break the "$O(L_N(1/2, c))$ barrier" one must make the numbers being tested for smoothness dramatically smaller. A key observation is that if the numbers are of size $O(L_N(2/3, c'))$ then they are $O(L_N(1/3, c''))$ smooth, for some constants c' and c'', with probability approximately $1/u^u = 1/L_N(1/3, c'/(3c'') + o(1))$. Hence, one can expect an algorithm

[7] The goal of Pollard's method was to factor integers of the form $n^3 + k$ where k is small. The the algorithm in the case of numbers of a special form is known as the **special number field sieve**.

with running time $O(L_N(1/3, c+o(1)))$ bit operations, for some constant c, by considering smaller values for smoothness.

It seems to be impossible to directly choose values x such that $x^2 \pmod N$ is of size $L_N(2/3, c + o(1))$ for some constant c. Hence, the number field sieve relies on two factor bases $\mathcal{B}_1$ and $\mathcal{B}_2$. Using smooth elements over $\mathcal{B}_1$ (respectively, $\mathcal{B}_2$) and linear algebra one finds an integer square u^2 and an algebraic integer square v^2. The construction allows us to associate an integer w modulo N to v such that $u^2 \equiv w^2 \pmod N$ and hence one can try to split N.

We briefly outline the ideas behind the algorithm. First, choose a monic irreducible polynomial $P(x) \in \mathbb{Z}[x]$ of degree d (where d grows like $\lfloor (3\log(N)/\log(\log(N)))^{1/3} \rfloor$) with a root $m = \lfloor N^{1/d} \rfloor$ modulo N (i.e., $P(m) \equiv 0 \pmod N$). Factor base $\mathcal{B}_1$ is primes up to $B = L_N(1/3, c)$ and factor base $\mathcal{B}_2$ is small prime ideals in the ring $\mathbb{Z}[\theta]$ in the number field $K = \mathbb{Q}(\theta) = \mathbb{Q}[x]/(P(x))$ (i.e., θ is a generic root of $P(x)$). The algorithm exploits, in the final step, the ring homomorphism $\phi : \mathbb{Z}[x]/(P(x)) \rightarrow \mathbb{Z}/N\mathbb{Z}$ given by $\phi(\theta) = m \pmod N$. Suppose the ideal $(a - b\theta)$ is a product of prime ideals in $\mathcal{B}_2$ (one factors the ideal $(a - b\theta)$ by factoring its norm in $\mathbb{Z}$), say

$$(a - b\theta) = \prod_{i=1}^{r} \wp_i^{e_i}.$$

Suppose also that $a - bm$ is a smooth integer in $\mathcal{B}_1$, say

$$a - bm = \prod_{j=1}^{s} p_j^{f_j}.$$

If these equations hold then we call $(a - b\theta)$ and $a - bm$ smooth and store a, b and the sequences of e_i and f_j. We do not call this a "relation" as there is no direct relationship between the prime ideals $\wp_i$ and the primes p_j. Indeed, the $\wp_j$ are typically non-principal ideals and do not necessarily contain an element of small norm. Hence, the two products are modelled as being "independent".

It is important to estimate the probability that both the ideal $(a - b\theta)$ and the integer $a - bm$ are smooth. One shows that taking integers $|a|, |b| \leq L_N(1/3, c' + o(1))$ for a suitable constant c' gives $(a - b\theta)$ of norm $L_N(2/3, c'' + o(1))$ and $a - bm$ of size $L_N(2/3, c''' + o(1))$ for certain constants c'' and c'''. To obtain a fast algorithm one uses sieving to determine within a range of values for a and b the pairs (a, b) such that both $a - bm$ and $(a - b\theta)$ factor over the appropriate factor base.

Performing linear algebra on both sides gives a set S of pairs (a, b) such that (ignoring issues with units and non-principal ideals)

$$\begin{aligned} \prod_{(a,b)\in S} (a - bm) &= u^2 \\ \prod_{(a,b)\in S} (a - b\theta) &= v^2 \end{aligned}$$

for some $u \in \mathbb{Z}$ and $v \in \mathbb{Z}[\theta]$. Finally we can "link" the two factor bases: Applying the ring homomorphism $\phi : \mathbb{Z}[\theta] \rightarrow \mathbb{Z}$ gives $u^2 \equiv \phi(v)^2 \pmod N$ and hence we have a chance to split N. A non-trivial task is computing the actual numbers u and $\phi(v)$ modulo N so that one can compute $\gcd(u - \phi(v), N)$.

Since one is only considering integers $a - bm$ in a certain range (and ideals in a certain range) for smoothness one relies on heuristic assumptions about the smoothness probability. The conjectural complexity of the number field sieve is $O(L_N(1/3, c + o(1)))$

bit operations as $N \to \infty$ where $c = (64/9)^{1/3} \approx 1.923$. Note, comparing with Exercise 15.2.12, that if $N \approx 2^{1024}$ then $L_N(1/3, 1.923) \approx 2^{87}$.

15.5 Index Calculus in Finite Fields

We now explain how similar ideas to the above have been used to find subexponential algorithms for the discrete logarithm problem in finite fields. The original idea is due to Kraitchik [354]. While all subexponential algorithms for the DLP share certain basic concepts, the specific details vary quite widely (in particular, precisely what "linear algebra" is required). We present in this section an algorithm that is very convenient when working in subgroups of prime order r in $\mathbb{F}_q^*$ as it relies only on linear algebra over the field $\mathbb{F}_r$.

Let $g \in \mathbb{F}_q^*$ have prime order r and let $h \in \langle g \rangle$. The starting point is the observation that if one can find integers $0 < Z_1, Z_2 < r$ such that

$$g^{Z_1} h^{Z_2} = 1 \tag{15.6}$$

in $\mathbb{F}_q^*$ then $\log_g(h) = -Z_1 Z_2^{-1} \pmod{r}$. The idea will be to find such a relation using a factor base and linear algebra. Such algorithms go under the general name of **index calculus** algorithms; the reason for this is that **index** is another word for discrete logarithm, and the construction of a solution to equation (15.6) is done by calculations using indices.

15.5.1 Rigorous Subexponential Discrete Logarithms Modulo p

We now sketch a subexponential algorithm for the discrete logarithm problem in $\mathbb{F}_p^*$. It is closely related to the random squares algorithm of Section 15.2. Let $g \in \mathbb{F}_p^*$ have order r (we will assume r is prime, but the general case is not significantly different) and let $h \in \mathbb{F}_p^*$ be such that $h \in \langle g \rangle$. We will also assume, for simplicity, that $r^2 \nmid (p-1)$ (we show in Exercise 15.5.8 that this condition can be avoided).

The natural idea is to choose the factor base $\mathcal{B}$ to be the primes in $\mathbb{Z}$ up to B. We let $s = \#\mathcal{B}$. One can take random powers $g^z \pmod{p}$ and try to factor over $\mathcal{B}$. One issue is that the values g^z only lie in a subgroup of $\mathbb{F}_p^*$ and so a strong smoothness heuristic would be required. To get a rigorous algorithm (under the assumption that $r^2 \nmid (p-1)$) write G' for the subgroup of $\mathbb{F}_p^*$ of order $(p-1)/r$, choose a random $\delta \in G'$ at each iteration and try to factor $g^z \delta \pmod{p}$; this is now a uniformly distributed element of $\mathbb{F}_p^*$ and so Corollary 15.1.8 can be applied. We remark that the primes p_i themselves do not necessarily lie in the subgroup $\langle g \rangle$.

Exercise 15.5.1. Let $r \mid (p-1)$ be a prime such that $r^2 \nmid (p-1)$. Let $g \in \mathbb{F}_p^*$ have order dividing r and denote by $G' \subseteq \mathbb{F}_p^*$ the subgroup of order $(p-1)/r$. Show that $\langle g \rangle \cap G' = \{1\}$.

Exercise 15.5.2. Give two ways to sample randomly from G'. When would each be used?
[Hint: see Section 11.4.]

The algorithm proceeds by choosing random values $1 \leq z < r$ and random $\delta \in G'$ and testing $g^z \delta \pmod{p}$ for smoothness. The i-th relation is

$$g^{z_i} \delta_i \equiv \prod_{j=1}^{s} p_j^{e_{i,j}} \pmod{p}. \tag{15.7}$$

The values z_i are stored in a vector and the values $\underline{e}_i = (e_{i,1}, \ldots, e_{i,s})$ are stored as a row in a matrix. We need s relations of this form. We also need at least one relation involving h (alternatively, we could have used a power of h in every relation in equation (15.7)) so try random values z_{s+1} and $\delta_{s+1} \in G'$ until $g^{z_{s+1}} h \delta_{s+1} \pmod p$ is B-smooth. One performs linear algebra modulo r to find integers $0 \leq \lambda_1, \ldots, \lambda_{s+1} < r$ such that

$$\sum_{i=1}^{s+1} \lambda_i \underline{e}_i = (rf_1, \ldots, rf_s) \equiv (0, \ldots, 0) \pmod r$$

where $f_1, \ldots, f_s \in \mathbb{Z}_{\geq 0}$. In matrix notation, writing $A = (e_{i,j})$, this is $(\lambda_1, \ldots, \lambda_{s+1})A \equiv (0, \ldots, 0) \pmod r$. In other words, the linear algebra problem is finding a non-trivial element in the kernel of the matrix A modulo r. Let $Z_1 = \sum_{i=1}^{s+1} \lambda_i z_i \pmod r$ and $Z_2 = \lambda_{s+1}$. Then

$$g^{Z_1} h^{Z_2} \left(\prod_i \delta_i^{\lambda_i} \right) \equiv \left(\prod_i p_i^{f_i} \right)^r \pmod p. \tag{15.8}$$

Since $g^{Z_1} h^{Z_2} \in \langle g \rangle$ and the other terms are all in G' it follows from Exercise 15.5.1 that $g^{Z_1} h^{Z_2} \equiv 1 \pmod r$ as required. We stress that it is not necessary to compute $\prod_i \delta_i^{\lambda_i}$ or the right hand side of equation (15.8).

The algorithm succeeds as long as $\lambda_{s+1} \not\equiv 0 \pmod r$ (and if $\lambda_{s+1} = 0$ then there is a linear dependence from the earlier relations, which can be removed by deleting one or more rows of the relation matrix).

Exercise 15.5.3. Show that if one replaces equation (15.7) by $g^{z_{1,i}} h^{z_{2,i}} \delta_i$ for random $z_{1,i}, z_{2,i}$ and δ_i then one obtains an algorithm that succeeds with probability $1 - 1/r$.

Example 15.5.4. Let $p = 223$. Then $g = 15$ has prime order $r = 37$. Suppose $h = 68$ is the instance of the DLP we want to solve. Let $\mathcal{B} = \{2, 3, 5, 7\}$. Choose the element $g_1 = 184$ of order $(p-1)/r = 6$. One can check that we have the following relations.

z	i	Factorization of $g^z g_1^i \pmod p$
1	1	$2^2 \cdot 3 \cdot 7$
33	0	$2^3 \cdot 7$
8	1	$3^2 \cdot 5$
7	0	$2^3 \cdot 3 \cdot 5$

One also finds the relation $hg^7 g_1^2 = 2^3 \cdot 3^2$.

We represent the relations as the vector and matrix

$$\underline{z} = \begin{pmatrix} 1 \\ 33 \\ 8 \\ 7 \\ 7 \end{pmatrix}, \qquad \begin{pmatrix} 2 & 1 & 0 & 1 \\ 3 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \\ 3 & 1 & 1 & 0 \\ 3 & 2 & 0 & 0 \end{pmatrix}.$$

Now perform linear algebra modulo 37. One finds the non-trivial kernel vector $\underline{v} = (1, 36, 20, 17, 8)$. Computing $Z_1 = \underline{v} \cdot \underline{z} = 7 \pmod{37}$ and $Z_2 = 8$ we find $g^{Z_1} h^{Z_2} \equiv 1 \pmod{223}$ and so the solution is $-Z_1 Z_2^{-1} \equiv 13 \pmod{37}$.

Exercise 15.5.5. Write the above algorithm in pseudocode (using trial division to determine the smooth relations).

Exercise 15.5.6. Let the notation be as above. Let T_B be the expected number of trials of random integers modulo p until one is B-smooth. Show that the expected running time of this algorithm (using naive trial division for the relations and using the Lanczos or Wiedemann methods for the linear algebra) is

$$O((\#\mathcal{B})^2 T_B M(\log(p)) + (\#\mathcal{B})^{2+o(1)} M(\log(r)))$$

bit operations as $p \to \infty$

Exercise 15.5.7. Show that taking $B = L_p(1/2, 1/2)$ is the optimal value to minimise the complexity of the above algorithm, giving a complexity of $O(L_p(1/2, 2 + o(1)))$ bit operations for the discrete logarithm problem in $\mathbb{F}_p^*$ as $p \to \infty$. (Note that, unlike many of the results in this chapter, this result does not rely on any heuristics.)

We remark that, in practice, rather than computing a full exponentiation g^z one might use a pseudorandom walk as done in Pollard rho. For further implementation tricks see Sections 5.1 to 5.5 of Odlyzko [469].

If g does not have prime order (e.g., suppose g is a generator of $\mathbb{F}_p^*$ and has order $p-1$) then there are several options: One can apply Pohlig-Hellman and reduce to subgroups of prime order and apply index calculus in each subgroup (or at least the ones of large order). Alternatively, one can apply the algorithm as above and perform the linear algebra modulo the order of g. There will usually be difficulties with non-invertible elements in the linear algebra, and there are several solutions, such as computing the Hermite normal form of the relation matrix or using the Chinese remainder theorem, we refer to Section 5.5.2 of Cohen [136] and Section 15.2.1 of Joux [317] for details.

Exercise 15.5.8. Give an algorithm similar to the above that works when $r^2 \mid (p-1)$.

Exercise 15.5.9. This exercise is about solving many different discrete logarithm instances $h_i = g^{a_i} \pmod p$, for $1 \leq i \leq n$, to the same base g. Once sufficiently many relations are found, determine the cost of solving each individual instance of the DLP. Hence show that one can solve any constant number of instances of the DLP to a given base $g \in \mathbb{F}_p^*$ in $O(L_p(1/2, 2 + o(1)))$ bit operations as $p \to \infty$.

15.5.2 Heuristic Algorithms for Discrete Logarithms Modulo p

To get a faster algorithm it is necessary to improve the time to find smooth relations. It is natural to seek methods to sieve rather than factoring each value by trial division, but it is not known how to do this for relations of the form in equation (15.7). It would also be natural to find an analogue to Pomerance's method of considering residues of size about the square-root of random; Exercise 15.5.10 gives an approach to this, but it does not lower the complexity.

Exercise 15.5.10. (Blake, Fuji-Hara, Mullin and Vanstone [61]) Once one has computed $w = g^z \delta \pmod p$ one can apply the Euclidean algorithm to find integers w_1, w_2 such that $w_1 w \equiv w_2 \pmod p$ and $w_1, w_2 \approx \sqrt{p}$. Since w_1 and w_2 are smaller one would hope that they are much more likely to both be smooth (however, note that both must be smooth). We now make the heuristic assumption that the probability each w_i is B-smooth is independent and the same as the probability that any integer of size $\sqrt{p}$ is B-smooth. Show that the heuristic running time of the algorithm has u^u replaced by $(u/2)^u$ (where $u = \log(p)/\log(B)$) and so the asymptotic running time remains the same.

Coppersmith, Odlyzko and Schroeppel [145] proposed an algorithm for the DLP in $\mathbb{F}_p^*$ that uses sieving. Their idea is to let $H = \lceil \sqrt{p} \rceil$ and define the factor base to be

$$\mathcal{B} = \{q : q \text{ prime}, q < L_p(1/2, 1/2)\} \cup \{H + c : 1 \leq c \leq L_p(1/2, 1/2 + \epsilon)\}.$$

Since $H^2 \pmod p$ is of size $\approx p^{1/2}$ it follows that if $(H + c_1), (H + c_2) \in \mathcal{B}$ then $(H + c_1)(H + c_2) \pmod p$ is of size $p^{1/2+o(1)}$. One can therefore generate relations in $\mathcal{B}$. Further, it is shown in Section 4 of [145] how to sieve over the choices for c_1 and c_2. A heuristic analysis of the algorithm gives complexity $L_p(1/2, 1 + o(1))$ bit operations.

The **number field sieve** (NFS) is an algorithm for the DLP in $\mathbb{F}_p^*$ with heuristic complexity $O(L_p(1/3, c + o(1)))$ bit operations. It is closely related to the number field sieve for factoring and requires algebraic number theory. As with the factoring algorithm, there are two factor bases. Introducing the DLP instance requires an extra algorithm (we will see an example of this in Section 15.5.4). We do not have space to give the details and instead refer to Schirokauer, Weber and Denny [519] or Schirokauer [515, 517] for details.

15.5.3 Discrete Logarithms in Small Characteristic

We now consider the discrete logarithm problem in $\mathbb{F}_q^*$ where $q = p^n$, p is relatively small (the case of most interest is $p = 2$) and n is large. We represent such a field with a polynomial basis as $\mathbb{F}_p[x]/(F(x))$ for some irreducible polynomial $F(x)$ of degree n. The natural notion of smoothness of an element $g(x) \in \mathbb{F}_p[x]/(F(x))$ is that it is a product of polynomials of small degree. Since factoring polynomials over finite fields is polynomial-time we expect to more easily get good algorithms in this case. The first work on this topic was due to Hellman and Reyneri but we follow Odlyzko's large paper [469]. First we quote some results on smooth polynomials.

Definition 15.5.11. Let p be prime and $n, b \in \mathbb{N}$. Let $I(n)$ be the number of monic irreducible polynomials in $\mathbb{F}_p[x]$ of degree n. A polynomial $g(x) \in \mathbb{F}_p[x]$ is called **b-smooth** if all its irreducible factors have degree $\leq b$. Let $N(n, b)$ be the number of b-smooth polynomials of degree exactly equal to n. Let $p(n, b)$ be the probability that a uniformly chosen polynomial of degree at most n is b-smooth.

Theorem 15.5.12. *Let p be prime and $n, b \in \mathbb{N}$.*

1. *$I(n) = \frac{1}{n} \sum_{d|n} \mu(d) p^{n/d} = \frac{1}{n} p^n + O(p^{n/2}/n)$ where $\mu(d)$ is the Möbius function.*[8]
2. *If $n^{1/100} \leq b \leq n^{99/100}$ then $N(n, b) = p^n (b/n)^{(1+o(1))n/b}$ as n tends to infinity.*
3. *If $n^{1/100} \leq b \leq n^{99/100}$ then $p(n, b)$ is at least $u^{-u(1+o(1))}$ where $u = n/b$ and n tends to infinity.*
4. *If $n^{1/100} \leq b \leq n^{99/100}$ then the expected number of trials before a randomly chosen element of $\mathbb{F}_p[x]$ of degree n is b-smooth is $u^{u(1+o(1))}$ as $u \to \infty$.*

Proof: Statement 1 follows from an elementary counting argument (see, for example, Theorem 3.25 of Lidl and Niederreiter [388]).

Statement 2 in the case $p = 2$ is Corollary A.2 of Odlyzko [469]. The general result was proved by Soundararajan (see Theorems 2.1 and 2.2 of Lovorn Bender and Pomerance [397]). Also see Section 9.15 of [388].

Statement 3 follows immediately from statement 2 and the fact there are p^n monic polynomials of degree at most n (when considering smoothness it is sufficient to study monic polynomials). Statement 4 follows immediately from statement 3. □

[8] This is the "prime number theorem for polynomials", $I(n) \approx p^n / \log_p(p^n)$.

The algorithm then follows exactly the ideas of the previous section. Suppose g has prime order $r \mid (p^n - 1)$ and $h \in \langle g \rangle$. The factor base is

$$\mathcal{B} = \{P(x) \in \mathbb{F}_p[x] : P(x) \text{ is monic, irreducible and } \deg(P(x)) \leq b\}$$

for some integer b to be determined later. Note that $\#\mathcal{B} = I(1) + I(2) + \cdots + I(b) \approx p^{b+1}/(b(p-1))$ (see Exercise 15.5.14). We compute random powers of g multiplied by a suitable $\delta \in G'$ (where, if $r^2 \nmid (p^n - 1)$, $G' \subseteq \mathbb{F}_{p^n}^*$ is the subgroup of order $(p^n - 1)/r$; when $r^2 \mid (p^n - 1)$ then use the method of Exercise 15.5.8), reduce to polynomials in $\mathbb{F}_p[x]$ of degree at most n, and try to factor them into products of polynomials from $\mathcal{B}$. By Exercise 2.12.11 the cost of factoring the b-smooth part of a polynomial of degree n is $O(bn\log(n)\log(p)M(\log(p^n))) = O(\log(p^n)^3)$ bit operations (in any case, polynomial-time). As previously, we are generating polynomials of degree n uniformly at random and so, by Theorem 15.5.12, the expected number of trials to get a relation is $u^{u(1+o(1))}$ where $u = n/b$ as $u \to \infty$. We need to obtain $\#\mathcal{B}$ relations in general. Then we obtain a single relation of the form $hg^a\delta = \prod_{P \in \mathcal{B}} P^{e_P}$, perform linear algebra, and hence solve the DLP.

Exercise 15.5.13. Write the above algorithm in pseudocode.

Exercise 15.5.14. Show that $\sum_{i=1}^{b} I(b) \leq \frac{1}{b}p^b(1 + 2/(p-1)) + O(bp^{b/2})$. Show that a very rough approximation is $p^{b+1}/(b(p-1))$.

Exercise 15.5.15. Let the notation be as above. Show that the complexity of this algorithm is at most

$$c_1 \#\mathcal{B} u^{u(1+o(1))} \log(q)^3 + c_2 (\#\mathcal{B})^{2+o(1)} M(\log(r))$$

bit operations (for some constants c_1 and c_2) as $n \to \infty$ in $q = p^n$.

For the complexity analysis it is natural to arrange that $\#\mathcal{B} \approx L_{p^n}(1/2, c)$ for a suitable constant c. Recall that $\#\mathcal{B} \approx p^b/b$. To have $p^b/b = L_{p^n}(1/2, c)$ then, taking logs,

$$b\log(p) - \log(b) = c\sqrt{n\log(p)(\log(n) + \log(\log(p)))}.$$

It follows that $b \approx c\sqrt{n\log(n)/\log(p)}$.

Exercise 15.5.16. Show that one can compute discrete logarithms in $\mathbb{F}_{p^n}^*$ in expected $O(L_{p^n}(1/2, \sqrt{2} + o(1)))$ bit operations for fixed p and as $n \to \infty$. (Note that this result does not rely on any heuristic assumptions.)

Exercise 15.5.17. Adapt the trick of exercise 15.5.10 to this algorithm. Explain that the complexity of the algorithm remains the same, but is now heuristic.

Lovorn Bender and Pomerance [397] give rigorous complexity $L_{p^n}(1/2, \sqrt{2} + o(1))$ bit operations as $p^n \to \infty$ and $p \leq n^{o(n)}$ (i.e., p is not fixed).

15.5.4 Coppersmith's Algorithm for the DLP in $\mathbb{F}_{2^n}^*$

This algorithm (inspired by the "systematic equations" of Blake, Fuji-Hara, Mullin and Vanstone [61]) was the first algorithm in computational number theory to have heuristic subexponential complexity of the form $L_q(1/3, c + o(1))$.

The method uses a polynomial basis for $\mathbb{F}_{2^n}$ of the form $\mathbb{F}_2[x]/(F(x))$ for $F(x) = x^n + F_1(x)$ where $F_1(x)$ has very small degree. For example, $\mathbb{F}_{2^{127}} = \mathbb{F}_2[x]/(x^{127} + x + 1)$.

The "systematic equations" of Blake et al are relations among elements of the factor base that come almost for free. For example, in $\mathbb{F}_{2^{127}}$, if $A(x) \in \mathbb{F}_2[x]$ is an irreducible polynomial in the factor base then $A(x)^{128} = A(x^{128}) \equiv A(x^2+x) \pmod{F(x)}$ and $A(x^2+x)$ is either irreducible or is a product $P(x)P(x+1)$ of irreducible polynomials of the same degree (Exercise 15.5.18). Hence, for many polynomials $A(x)$ in the factor base one gets a non-trivial relation.

Exercise 15.5.18. Let $A(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial. Show that $A(x^2+x)$ is either irreducible or a product of two polynomials of the same degree.

Coppersmith [140] extended the idea as follows: Let $b \in \mathbb{N}$ be such that $b = cn^{1/3}\log(n)^{2/3}$ for a suitable constant c (later we take $c = (2/(3\log(2)))^{2/3}$), let $k \in \mathbb{N}$ be such that $2^k \approx \sqrt{n/b} \approx \frac{1}{\sqrt{c}}(n/\log(n))^{1/3}$, and let $l = \lceil n/2^k \rceil \approx \sqrt{nb} \approx \sqrt{c}n^{2/3}\log(n)^{1/3}$. Let $\mathcal{B} = \{A(x) \in \mathbb{F}_2[x] : \deg(A(x)) \leq b, A(x) \text{ irreducible}\}$. Note that $\#\mathcal{B} \approx 2^b/b$ by Exercise 15.5.14. Suppose $A(x), B(x) \in \mathbb{F}_2[x]$ are such that $\deg(A(x)) = d_A \approx b$ and $\deg(B(x)) = d_B \approx b$ and define $C(x) = A(x)x^l + B(x)$. In practice one restricts to pairs $(A(x), B(x))$ such that $\gcd(A(x), B(x)) = 1$. The crucial observation is that

$$C(x)^{2^k} = A(x^{2^k}) \cdot (x^{2^k})^l + B(x^{2^k}) \equiv A(x^{2^k})x^{2^k l - n}F_1(x) + B(x^{2^k}) \pmod{F(x)}. \quad (15.9)$$

Write $D(x)$ for the right hand side of equation (15.9). We have $\deg(C(x)) \leq \max\{d_A + l, d_B\} \approx l \approx n^{2/3}\log(n)^{1/3}$ and $\deg(D(x)) \leq \max\{2^k d_A + (2^k l - n) + \deg(F_1(x)), 2^k d_B\} \approx 2^k b \approx n^{2/3}\log(n)^{1/3}$.

Example 15.5.19. (Thomé [608]) Let $n = 607$ and $F_1(x) = x^9 + x^7 + x^6 + x^3 + x + 1$. Let $b = 23$, $d_A = 21, d_B = 28$, $2^k = 4$, $l = 152$. The degrees of $C(x)$ and $D(x)$ are 173 and 112 respectively.

We have two polynomials $C(x), D(x)$ of degree $\approx n^{2/3}$ that we wish to be b-smooth where $b \approx n^{1/3}\log(n)^{2/3}$. We will sketch the complexity later under the heuristic assumption that, from the point of view of smoothness, these polynomials are independent. We will also assume that the resulting relations are essentially random (and so with high probability there is a non-trivial linear dependence once $\#\mathcal{B}+1$ relations have been collected).

Having generated enough relations among elements of the factor base, it is necessary to find some relations involving the elements g and h of the DLP instance. This is not trivial. All DLP algorithms having complexity $L_q(1/3, c+o(1))$ feature a process called **special q-descent** that achieves this. The first step is to express g (respectively, h) as a product $\prod_i G_i(x)$ of polynomials of degree at most $b_1 = n^{2/3}\log(n)^{1/3}$; this can be done by multiplying g (resp. h) by random combinations of elements of $\mathcal{B}$ and factoring (one can also apply the Blake et al trick as in Exercise 15.5.10). We now have a list of around $2n^{1/3} < n$ polynomials $G_i(x)$ of degree $\approx n^{2/3}$ that need to be "smoothed" further. Section VII of [140] gives a method to do this: essentially one performs the same sieving as earlier except that $A(x)$ and $B(x)$ are chosen so that $G_i(x) \mid C(x) = A(x)x^l + B(x)$ (not necessarily with the same value of l or the same degrees for $A(x)$ and $B(x)$). Defining $D(x) = C(x)^{2^k} \pmod{F(x)}$ (not necessarily the same value of k as before) one hopes that $C(x)/G(x)$ and $D(x)$ are b-smooth. After sufficiently many trials one has a relation that expresses $G_i(x)$ in terms of elements of $\mathcal{B}$. Repeating for the polynomially many values $G_i(x)$ one eventually has the values g and h expressed in terms of elements of $\mathcal{B}$. One can then do linear algebra modulo the order of g to find integers Z_1, Z_2 such that $g^{Z_1}h^{Z_2} = 1$ and the DLP is solved.

Example 15.5.20. We give an example of Coppersmith's method for $\mathbb{F}_{2^{15}} = \mathbb{F}_2[x]/(F(x))$ where $F(x) = x^{15} + x + 1$. We consider the subgroup of $\mathbb{F}_{2^{15}}^*$ of order $r = 151$ (note that $(2^{15} - 1)/r = 7 \cdot 31 = 217$). Let $g = x^{11} + x^7 + x^5 + x^2 + 1$ and $h = x^{14} + x^{11} + x^{10} + x^9 + 1$ be the DLP instance.

First note that $n^{1/3} \approx 2.5$ and $n^{2/3} \approx 6.1$. We choose $b = 3$ and so $\mathcal{B} = \{x, x+1, x^2 + x + 1, x^3 + x + 1, x^3 + x^2 + 1\}$. We hope to be testing polynomials of degree around 6 to 8 for smoothness.

First, we find some "systematic equations". We obviously have the relation $x^{15} = x+1$. We also have $(x + 1)^{16} = x^2 + x + 1$ and $(x^3 + x + 1)^{16} = (x^3 + x + 1)(x^3 + x^2 + 1)$.

Now, we do Coppersmith's method. We must choose $2^k \approx \sqrt{n/b} = \sqrt{5} \approx 2.2$ so take $2^k = 2$. Let $l = \lceil n/2^k \rceil = 8$, choose $A(x)$ and $B(x)$ of degree at most 2, set $C(x) = A(x)x^8 + B(x)$ and $D(x) = C(x)^2 \pmod{F(x)}$, and test $C(x)$ and $D(x)$ for smoothness over $\mathcal{B}$. We find the following pairs $(A(x), B(x))$ such that both $C(x)$ and $D(x)$ factor over $\mathcal{B}$.

$A(x)$	$B(x)$	$C(x)$	$D(x)$
1	1	$(x+1)^8$	$x^2 + x + 1$
1	x	$x(x+1)(x^3 + x + 1)(x^3 + x^2 + 1)$	x
1	x^2	$x^2(x+1)^2(x^2 + x + 1)^2$	$x(x^3 + x + 1)$

The first relation in the table is a restatement of $(x + 1)^{16} = x^2 + x + 1$. All together, we have the relation matrix

$$\begin{pmatrix} 15 & -1 & 0 & 0 & 0 \\ 0 & 16 & -1 & 0 & 0 \\ 0 & 0 & 0 & 15 & -1 \\ 1 & 2 & 0 & 2 & 2 \\ 3 & 4 & 4 & -1 & 0 \end{pmatrix}. \tag{15.10}$$

To solve the DLP one can now try to express g and h over the factor base. One has

$$g^{22} = x(x + 1)(x^2 + x + 1)^2(x^3 + x^2 + 1).$$

For h we find

$$hg^{30} = x^6(x + 1)^4 G(x)$$

where $G(x) = x^4 + x + 1$ is a "large prime". To "smooth" $G(x)$ we choose $A(x) = 1$, $B(x) = A(x)x^8 \pmod{G(x)} = x^2 + 1$, $C(x) = A(x)x^8 + B(x)$ and $D(x) = C(x)^2 \pmod{F(x)}$. One finds $C(x) = G(x)^2$ and $D(x) = (x + 1)(x^3 + x^2 + 1)$. In other words, $G(x)^4 = (x + 1)(x^3 + x^2 + 1)$.

There are now two ways to proceed. Following the algorithm description above we add to the matrix the two rows $(1, 1, 2, 0, 1)$ and $4(6, 4, 0, 0, 0) + (0, 1, 0, 0, 0, 1) = (24, 17, 0, 0, 1)$ corresponding to g^{22} and $h^4 g^{120}$. Finding a non-trivial kernel vector modulo 151, such as $(1, 114, 0, 132, 113, 133, 56)$ gives the relation

$$1 = (g^{22})^{133}(h^4 g^{120})^{56} = g^{133} h^{73}$$

from which we deduce $h = g^{23}$.

An alternative approach to the linear algebra is to diagonalise the system in equation (15.10) using linear algebra over $\mathbb{Z}$ (or at least modulo $2^{15} - 1$) to get $x + 1 = x^{15}, x^2 + x + 1 = x^{240}, x^3 + x + 1 = x^{1023}$ and $x^3 + x^2 + 1 = x^{15345}$. One then gets

$$g^{22} = x(x + 1)(x^2 + x + 1)^2(x^3 + x^2 + 1) = x^{1+15+2\cdot 240+15345} = x^{15841}$$

and so

$$g = x^{15841 \cdot 22^{-1} \pmod{(2^{15}-1)}} = x^{26040} = (x^{217})^{120}.$$

Similarly, $G(x)^4 = (x+1)(x^3+x^2+1) = x^{15+15345} = x^{15360}$ and so $G(x) = x^{3840}$. Finally,

$$h = g^{-30}x^6(x+1)^4G(x) = x^{-30 \cdot 26040+6+4 \cdot 15+3840} = x^{9114} = (x^{217})^{42}$$

and so $h = g^{42 \cdot 120^{-1} \pmod{151}} = g^{23}$.

Conjecture 15.5.21. *Coppersmith's algorithm solves the DLP in $\mathbb{F}_q^*$ where $q = 2^n$ in $L_q(1/3, (32/9)^{1/3} + o(1))$ bit operations as $n \to \infty$.*

Note that, to compare with Exercise 15.2.12, if $q = 2^{1024}$ then $L_q(1/3, (32/9)^{1/3}) \approx 2^{67}$.

This conjecture would hold if the probability that the polynomials $C(x)$ and $D(x)$ are smooth was the same as for independently random polynomials of the same degree. We now give a justification for the constant. Let $b = cn^{1/3}\log(n)^{2/3}$. Note that $2^k \approx \sqrt{n/b} \approx (n/\log(n))^{1/3}/\sqrt{c}$ and $l \approx \sqrt{nb}$. We need around $2^b/b$ relations, and note that $\log(2^b/b) \approx b\log(2) = c\log(2)n^{1/3}\log(n)^{2/3}$. We have $\deg(C(x)) \approx d_A + l$ and $\deg(D(x)) \approx 2^k d_A$. The number of trials until $C(x)$ is b-smooth is u^u where $u = (d_A + l)/b \approx h/b \approx \sqrt{n/b} = \frac{1}{\sqrt{c}}(n/\log(n))^{1/3}$. Hence, $\log(u^u) = u\log(u) \approx \frac{1}{3\sqrt{c}}n^{1/3}\log(n)^{2/3}$. Similarly, the number of trials until $D(x)$ is b-smooth is approximately u^u where $u = (2^k d_A)/b \approx 2^k \approx \sqrt{n/b}$ and the same argument applies. Since both events must occur the expected number of trials to get a relation is $\exp(\frac{2}{3\sqrt{c}}(n\log(n)^2)^{1/3})$. Hence, total expected time to generate enough relations is

$$\exp\left((c\log(2) + \tfrac{2}{3\sqrt{c}})n^{1/3}\log(n)^{2/3}\right).$$

This is optimised when $c^{3/2}\log(2) = 2/3$, which leads to the stated complexity for the first stage of the algorithm. (In practice one chooses c so that there are enough smooth pairs $(C(x), D(x))$ to generate the required number of relations.) The linear algebra is $O((2^b/b)^{2+o(1)}M(\log(r)))$ bit operations, which is the same complexity, and the final stage of solving the DLP has lower complexity (it is roughly the same as the cost of finding polynomially many smooth relations, rather than finding $2^b/b$ of them). For more details about the complexity of Coppersmith's method we refer to Section 2.4 of Thomé [608].

Since one can detect smoothness of polynomials in polynomial-time it is not necessary, from a complexity theory point of view, to sieve. However, in practice sieving can be worthwhile and a method to do this was given by Gordon and McCurley [263].

Coppersmith's idea is a special case of a more general approach to index calculus algorithms known as the **function field sieve**. Note that Coppersmith's algorithm only has one factor base, whereas the function field sieve works using two factor bases.

15.5.5 The Joux-Lercier Algorithm

The function field sieve of Adleman is a general algorithm for discrete logarithms in $\mathbb{F}_{p^n}$ where p is relatively small compared with n. Joux and Lercier gave a much simpler and better algorithm. We will sketch this algorithm, but refer to Joux and Lercier [318] and Section 15.2 of [317] for full details. We also refer to [517] for a survey of the function field sieve.

Let p be prime and $n \in \mathbb{N}$. Let $d = \lceil \sqrt{n} \rceil$. Suppose one has monic polynomials $F_1(t), F_2(t) \in \mathbb{F}_p[t]$ such that $\deg(F_1(t)) = \deg(F_2(t)) = d$ and $F_2(F_1(t)) - t$ has an irreducible factor $F(t)$ of degree n. We represent $\mathbb{F}_{p^n}$ with the polynomial basis $\mathbb{F}_p[t]/(F(t))$.

Given a prime p and an integer n one can find such polynomials $F_1(t)$ and $F_2(t)$ in very little time (e.g., by choosing polynomials of the right degree uniformly at random and testing the condition using polynomial factorisation).

Exercise 15.5.22. Let $n = 15$. Find polynomials $F_1(t), F_2(t) \in \mathbb{F}_2[t]$ of degree 4 such that $F_2(F_1(t)) - t$ has an irreducible factor of degree 15.

Now consider the polynomial ring $A = \mathbb{F}_p[x, y]$ and two ring homomorphisms $\psi_1 : A \to A_1 = \mathbb{F}_p[x]$ by $\psi_1(y) = F_1(x)$ and $\psi_2 : A \to A_2 = \mathbb{F}_p[y]$ by $\psi_2(x) = F_2(y)$. Define $\phi_1 : A_1 \to \mathbb{F}_{p^n}$ by $\phi_1(x) = t \pmod{F(t)}$ and $\phi_2 : A_2 \to \mathbb{F}_{p^n}$ by $\phi_2(y) = F_1(t) \pmod{F(t)}$.

Exercise 15.5.23. Let the notation be as above and $G(x, y) \in \mathbb{F}_p[x, y]$. Show that $\phi_1(\psi_1(G(x, y))) = \phi_2(\psi_2(G(x, y)))$ in $\mathbb{F}_{p^n}$.

Let $\mathcal{B}_1 \subseteq A_1 = \mathbb{F}_p[x]$ and $\mathcal{B}_2 \subseteq \mathbb{F}_p[y]$ be the sets of linear polynomials. The idea of the algorithm is simply to consider polynomials in $\mathbb{F}_p[x, y]$ of the form $G(x, y) = xy + ax + by + c$. If $\psi_1(G(x, y)) = (x + b)F_1(x) + (ax + c)$ factors over $\mathcal{B}_1$ as $\prod_{i=1}^{d+1}(x - u_i)$ and if $\psi_2(G(x, y)) = (y + a)F_2(y) + (by + c)$ factors over $\mathcal{B}_2$ as $\prod_{j=1}^{d+1}(y - v_j)$ then we have a relation. The point is that such a relation corresponds to

$$\prod_{i=1}^{d+1}(t - u_i) = \prod_{j=1}^{d+1}(F_1(t) - v_j)$$

in $\mathbb{F}_{p^n}$.

One also needs to introduce the DLP instance by using a special q-descent: given an irreducible polynomial $q(x)$ one constructs polynomials $a(x), b(x)$ such that $q(x) \mid (a(x)F_1(x) + b(x))$ and one hopes that $(a(x)F_1(x) + b(x))/q(x)$ has small factors and that $a(F_2(y))y + b(F_2(y))$ has small factors, and hence iterate the process. When enough relations are collected (including at least one "systematic equation" to remove the parasitic solution explained on page 442 of Joux/indexAJoux, A. [317]) one can perform linear algebra to solve the DLP. The heuristic complexity of this algorithm is shown in [318] and Section 15.2.1.2 of [317] to be between $L_{p^n}(1/3, 3^{1/3} + o(1))$ and $L_{p^n}(1/3, (32/9)^{1/3} + o(1))$ for $p \leq L_{p^n}(1/3, (4/9)^{1/3} + o(1))$.

15.5.6 Number Field Sieve for the DLP

Concepts from the number field sieve for factoring have been applied in the setting of the DLP. Again, one uses two factor bases, corresponding to ideals in the ring of integers of some number field (one of the number fields may be $\mathbb{Q}$). As with Coppersmith's method, once sufficiently many relations have been found among elements of the factor bases, special q-descent is needed to solve a general instance of the DLP. We refer to Schirokauer [517] for details of the NFS algorithm for the DLP, and also for the heuristic arguments that one can solve the DLP in $\mathbb{F}_p^*$ in $L_p(1/3, (64/9)^{1/3} + o(1))$ bit operations. When p has a special form (e.g., $p = 2^n \pm 1$) then the **special number field sieve** (SNFS) can be used to solve the DLP in (heuristic) $L_p(1/3, (32/9)^{1/3} + o(1))$ bit operations, see [518].

We should also mention the **special function field sieve** (SFFS) for solving the DLP in $\mathbb{F}_{p^n}^*$, which has heuristic complexity $L_{p^n}(1/3, (32/9)^{1/3} + o(1))$ bit operations as $p^n \to \infty$ as long as $p \leq n^{o(\sqrt{n})}$, see Schirokauer [516, 517].

15.5.7 Discrete Logarithms for all Finite Fields

We have sketched algorithms for the DLP in $\mathbb{F}_p^*$ when p is large or $\mathbb{F}_{p^n}^*$ when p is relatively small. We have not considered cases $\mathbb{F}_q^*$ where $q = p^n$ with p large and $n > 1$. The basic concepts can be extended to cover all cases, but ensuring that subexponential complexity is achieved for all combinations of p and n is non-trivial. Adleman and Demarrais [2] were the first to give a heuristic subexponential algorithm for all finite fields. They split the problem space into $p > n$ and $p \leq n$; in the latter case they have complexity $L_q(1/2, 3 + o(1))$ bit operations as $q \to \infty$ and in the former case heuristic complexity $L_q(1/2, c + o(1))$ for a non-explicit constant c.

Heuristic algorithms with complexity $L_q(1/3, c + o(1))$ for all finite fields are given by Joux and Lercier [318] and Joux, Lercier, Smart and Vercauteren [319].

15.6 Discrete Logarithms on Hyperelliptic Curves

Some index calculus algorithms for the discrete logarithm problem in finite fields generalise naturally to solving the DLP in the divisor class group of a curve. Indeed, some of these algorithms also apply to the ideal class group of a number field, but we do not explore that situation in this book. An excellent survey of discrete logarithm algorithms for divisor class groups is Chapter VII of [65].

We consider hyperelliptic curves $C : y^2 + H(x)y = F(x)$ over $\mathbb{F}_q$ of genus g, so $\deg(H(x)) \leq g + 1$ and $\deg(F(x)) \leq 2g + 2$. Recall that elements of the divisor class group have a Mumford representation $(u(x), y - v(x))$ (for curves with a split model there is also an integer $0 \leq n \leq g - \deg(u(x))$ to take into account the behaviour at infinity). Let D_1 and D_2 be reduced divisors representing divisor classes of order r (where r is a prime such that $r^2 \nmid \#\mathrm{Pic}^0_{\mathbb{F}_q}(C)$). The goal is to compute $a \in \mathbb{Z}/r\mathbb{Z}$ such that $D_2 \equiv [a]D_1$.

Recall from Exercise 10.3.12 that a reduced divisor with Mumford representation $(u(x), v(x))$ is said to be a **prime divisor** if the polynomial $u(x)$ is irreducible over $\mathbb{F}_q$. The **degree** of the effective affine divisor is $\deg(u(x))$. Any effective affine divisor D can be written as a sum of prime effective affine divisors by factoring the $u(x)$ polynomial of its Mumford representation. Hence, it is natural to define D to be b-smooth if it is a sum of prime effective divisors of degree at most b. This suggests selecting the factor base $\mathcal{B}$ to consist of all prime effective affine divisors of degree at most b for some smoothness bound $1 \leq b \leq g$.

We assume that $\mathcal{B}$ generates the group $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$; this is immediate when the group has prime order and $\mathcal{B}$ contains a non-trivial element. Voloch [624] has proved that degree 1 primes generate $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ whenever $q > (8g(C) - 2)^2$, where $g(C)$ is the genus of C.

One can obtain an algorithm for the DLP of a familiar form, by generating reduced divisors and testing whether they are smooth. One issue is that our smoothness results for polynomials apply when polynomials are sampled uniformly from the set of all polynomials of degree n in $\mathbb{F}_q[x]$, whereas we now need to apply the results to the set of polynomials $u(x) \in \mathbb{F}_q[x]$ of degree g that arise in Mumford's representation. This issue is handled using Theorem 15.6.1.

There are two rather different ways to generate reduced divisors, both of which are useful for the algorithm.

1. One can take random group elements of the form $[n]D_1$ or $[n_1]D_1 + [n_2]D_2$ and compute the Mumford representation of the corresponding reduced effective affine divisor. This is the same approach as used in Section 15.5.1 and, in the context of ideal/divisor class groups, is sometimes called the **Hafner-McCurley algorithm**.

If the divisor is $\mathcal{B}$-smooth then we obtain a relation between elements of $\mathcal{B}$ and D_1 and D_2.

2. One can consider the effective affine divisor of the function $a(x) + yb(x)$ for random polynomials $a(x), b(x)$. This idea is due to Adleman, DeMarrais and Huang [4]. Since a principal divisor is equivalent to zero in the ideal class group, if the divisor is $\mathcal{B}$-smooth then we get a relation in $\mathcal{B}$.

To introduce the instance of the DLP into the system it is necessary to have some relations involving D_1 and D_2. This can either be done using the first method, or by choosing $a(x)$ and $b(x)$ so that points in the support of either D_1 or D_2 lie in the support of $\mathrm{div}(a(x)+yb(x))$ (we have seen this kind of idea already, e.g., in Coppersmith's algorithm).

It is convenient to add to $\mathcal{B}$ all points at infinity and all points $P \in C(\overline{\mathbb{F}}_q)$ such that $P = \iota(P)$ (equivalently all $\mathbb{F}_q$-rational prime divisors with this property). Since the latter divisors all have order 2 one automatically obtains relations that can be used to eliminate them during the linear algebra stage of the algorithm. Hence, we say that a reduced divisor $D = \mathrm{div}(u(x), y - v(x))$ in Mumford representation is b-**smooth** if $u(x)$ is b-smooth after any factors corresponding to points of order 2 have been removed.

Let C be a hyperelliptic curve over $\mathbb{F}_q$ of genus g and $1 \leq b < g$. Prime effective affine divisors on C of degree b correspond to irreducible polynomials $u(x)$ of degree b (and for roughly half of all such polynomials $u(x)$ there are two solutions $v(x)$ to $v(x)^2 + v(x)H(x) - F(x) \equiv 0 \pmod{u(x)}$). Hence, it is natural to expect that there are approximately q^b/b such divisors. It follows that $\#\mathcal{B}$ should be around $\sum_{i=1}^{b} q^i/i \approx \frac{1}{b}p^b(1 + 2/(p-1))$ by the same argument as Exercise 15.5.14.

For the analysis, one needs to estimate the probability that a randomly chosen reduced divisor is smooth.

Theorem 15.6.1. *(Theorem 6 of Enge and Stein [198]) Let C be a hyperelliptic curve of genus g over $\mathbb{F}_q$. Let $c > 1$ and let $b = \lceil \log_q(L_{q^g}(1/2, c)) \rceil$. Then the number of b-smooth reduced divisors of degree g is at least*

$$\frac{q^g}{L_{q^g}(1/2, 1/(2c) + o(1))}$$

for fixed q and $g \to \infty$.

Note that the smoothness bound in the above result is the ceiling of a real number. Hence one cannot deduce subexponential running time unless the genus is sufficiently large compared with the field size.

15.6.1 Index Calculus on Hyperelliptic Curves

Suppose that $r \mid N = \#\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ and $r^2 \nmid N$. Suppose $\overline{D}_1, \overline{D}_2$ are two divisor classes on C over $\mathbb{F}_q$ of order r represented by reduced divisors D_1 and D_2. The algorithm of Section 15.5.1 immediately applies to solve the DLP: choose the factor base as above; generate random reduced divisors by computing $[n_1]D_1 + [n_2]D_2 + \delta$ (where δ is uniformly chosen[9] from the subgroup $G' \subseteq \mathrm{Pic}^0_{\mathbb{F}_q}(C)$ of order N/r); store the resulting smooth relations; perform linear algebra modulo r to find integers a, b such that $[a]D_1 + [b]D_2 \equiv 0$ (extra care is needed when there are two points at infinity to be sure the relation is correct).

[9] We assume that generators for this group are known so that it is easy to sample uniformly from this group.

Exercise 15.6.2. Show that the expected running time of this algorithm is (rigorously!) $L_{q^g}(1/2, \sqrt{2} + o(1))$ bit operations as $g \to \infty$.

We refer to Section VII.5 of [65] for practical details of the algorithm. Note that the performance can be improved using the sieving method of Flassenberg and Paulus [206].

15.6.2 The Algorithm of Adleman, De Marrais and Huang

This algorithm, from [4], uses the same factor base as the method of the previous section. The main difference is to generate relations by decomposing principal divisors $A(x) + yB(x)$. An advantage of this approach is that group operations are not required.

By Exercise 10.1.26 it is easy to compute $v_P(A(x) + yB(x))$ by computing the norm $A(x)^2 - H(x)A(x)B(x) - F(x)B(x)^2$ and factoring it as a polynomial. If $\deg(A(x)) = d_A < g$ and $\deg(B(x)) = d_B < g$ then the norm has degree at most $\max\{2d_A, (g+1) + d_A + d_B, 2g + 2 + 2d_B\}$, which is much larger in general than the degree g polynomial in a reduced Mumford representation, but still $O(g)$ in practice.

We need to make the heuristic assumption that the probability the norm is b-smooth is the same as the probability that a random polynomial of the same degree is b-smooth. We therefore assume the expected number of trials to get an $L_{q^g}(1/2, c)$-smooth polynomial is $L_{q^g}(1/2, 1/(2c) + o(1))$ as g tends to infinity.

We also need some relations involving D_1 and D_2. Adleman et al do this by first decomposing D_1 and D_2 as a sum of prime divisors. Then they "smooth" each prime divisor $\mathrm{div}(u(x), y - v(x))$ by choosing polynomials $B(x), W(x) \in \mathbb{F}_q[x]$, setting $A'(x) = B(x)(v(x) + H(x)) \pmod{u(x)}$ and then $A(x) = A'(x) + u(x)W(x)$. One computes $N(x) = (A(x)^2 - H(x)A(x)B(x) - F(x)B(x)^2)$. By construction, $u(x) \mid N(x)$ and one continues randomly choosing A and W until $N(x)/u(x)$ is b-smooth.

The details of the algorithm are then the same as the algorithm in Section 15.5.1: one uses linear algebra modulo r to get a relation $[a]D_1 + [b]D_2 \equiv 0$ (again, care is needed when there are two points at infinity). We leave the details as an exercise.

Exercise 15.6.3. Write pseudocode for the Adleman, DeMarrais, Huang algorithm.

The heuristic complexity of the algorithm is of the same form as the earlier algorithm (the cost of smoothing the divisors D_1 and D_2 is heuristically the same as finding less than $2g$ relations so is negligible. One obtains heuristic asymptotic complexity of $L_{q^g}(1/2, \sqrt{2} + o(1))$ bit operations as g tends to infinity. This is much better than the complexity claimed in [4] since that paper also gives an algorithm to compute the group structure (and so the linear algebra requires computing the Hermite normal form).

These ideas will be used again in Section 15.9.1.

15.6.3 Gaudry's Algorithm

Gaudry [242] considered the algorithm of Section 15.6.1 for fixed genus, rather than asympotically as $g \to \infty$. In particular he chose the smoothness bound $b = 1$ (so the factor base $\mathcal{B}$ only consists of degree one prime divisors, i.e., points). Good surveys of Gaudry's algorithm are given in Chapter VII of [65] and Section 21.2 of [16].

Exercise 15.6.4. Let C be a hyperelliptic curve of genus g over a finite field $\mathbb{F}_q$. Show that the number of prime divisors on C of degree 1 is $\#C(\mathbb{F}_q) = q(1 + o(1))$ for fixed g as $q \to \infty$. Hence, show that the probability that a randomly chosen reduced divisor is 1-smooth is $\frac{1}{g!}(1 + o(1))$ as $q \to \infty$.

Exercise 15.6.5. Following Exercise 15.6.4, it is natural to conjecture that one needs to choose $O(g!q(1+o(1)))$ divisors (again, this is for fixed g as $q \to \infty$, in which case it is more common to write it as $O(q(1+o(1)))$) to find enough relations to have a non-trivial linear dependence in $\mathcal{B}$. Under this assumption, show that the heuristic expected running time of Gaudry's algorithm is at most

$$c_1 g^2 g! q(1+o(1))M(\log(q)) + c_2 g^3 q^2 M(\log(q))) = O(q^2 M(\log(q))(1+o(1))) \quad (15.11)$$

bit operations (for some constants c_1 and c_2) for fixed g as $q \to \infty$.

The first term in equation (15.11) is the running time for relation generation. If g is fixed then asymptotically this is dominated by the second term, which is the running time for the linear algebra stage. If g is fixed, then the running time is $\tilde{O}(q^2)$ bit operations. Hence Gaudry's algorithm is asymptotically faster than Pollard's rho method for hyperelliptic curves of a fixed genus $g \geq 5$. However, the hidden constant in the expression $\tilde{O}(q^2)$ depends very badly on g. In practice, Gaudry's method seems to be superior to rho for small g (e.g., $g = 5, 6, 7$).

Harley and Thériault (see [607]) suggested reducing the factor base size in Gaudry's algorithm in order to balance the running times of the relation generation and linear algebra stages. Thériault [607] also proposed a "large prime" variant of Gaudry's algorithm. Gaudry, Thériault, Thomé and Diem [250] proposed a "double large prime" variant of Gaudry's algorithm that is based on the double large prime strategy that was successful in accelerating integer factorization algorithms. The factor base $\mathcal{B}$ is now chosen to be a subset of the degree one divisors and degree one divisors that are not in $\mathcal{B}$ are called *large primes*. A divisor is defined to be smooth if it can be written as a sum of prime divisors and at most two large primes. Relations are collected as before, and then combined to eliminate the large primes (we refer to Section 21.3 of [16] for further discussion of large primes and graph methods for eliminating them). It is shown in [250] that, for fixed g, the expected running time of the algorithm is $\tilde{O}(q^{2-\frac{2}{g}})$ bit operations. This is faster than Pollard rho for $g \geq 3$ when q is sufficiently large. Gaudry's approach was generalised to all curves of fixed genus by Diem [176].

15.7 Weil Descent

As we have seen, there are subexponential algorithms for the DLP in the divisor class group of a hyperelliptic curve of high genus. A natural approach to solve the DLP on elliptic curves is therefore to transform the problem into a DLP on a high genus curve. However, the naive way to do this embeds a small problem into a big one, and does not help to solve the DLP. Frey [212] proposed[10] to use Weil restriction of scalars to transform the DLP on an elliptic curve $E(\mathbb{F}_{q^n})$ for $n > 1$ to the DLP on a curve of genus $g \geq n$ over $\mathbb{F}_q$. Frey called this idea **Weil descent**.

Geometrically the principle is to identify the Weil restriction of an open affine subset of $E(\mathbb{F}_{q^n})$ (see Section 5.7) with an open affine subset of an Abelian variety A over $\mathbb{F}_q$ of dimension n. One can then try to find a curve C on A, so that there is a map from the Jacobian of C to A. Following Gaudry, Hess and Smart [246] it is more convenient to express the situation in terms of function fields and divisor class groups. We only sketch

[10] The standard reference is a lecture given by Frey at the ECC 1998 conference. His talk was mostly about a different (constructive) application of Weil restriction of scalars. However, he did mention the possibility of using this idea for an attack. Galbraith and Smart developed the details further in [229] and many works followed.

the details since an excellent survey is provided by Hess in Chapter VIII of [65] and many important details are explained by Diem in [172].

Let E be an elliptic curve over $\mathbb{K} = \mathbb{F}_{q^n}$ and let $\Bbbk = \mathbb{F}_q$. The function field of E is $\mathbb{K}(E)$. The idea (called in this setting a **covering attack**) is to find a curve C over $\mathbb{K}$ such that $\mathbb{K}(C)$ is a finite extension of $\mathbb{K}(E)$ (so that there is a map $C \to E$ of finite degree) and such that there is an automorphism σ of degree n on $\mathbb{K}(C)$ extending the q-power Frobenius so that the fixed field of $\mathbb{K}(C)$ under $\langle\sigma\rangle$ is $\Bbbk(C^0)$ for some curve C^0. The composition of the conorm map from $E(\mathbb{K})$ to $\mathrm{Pic}^0_C(\mathbb{K})$ and the norm map from $\mathrm{Pic}^0_C(\mathbb{K})$ to $\mathrm{Pic}^0_{C^0}(\Bbbk)$ transfers the DLP from $E(\mathbb{K})$ to $\mathrm{Pic}^0_{C^0}(\Bbbk)$. Hence, as long as the composition of these maps is not trivial, then one has reduced the DLP from $E(\mathbb{K})$ to the divisor class group of a curve C^0 over $\Bbbk$. One can then solve the DLP using an index calculus algorithm, which is feasible if the genus of C^0 is not too large.

A variant of the Weil descent concept that avoids function fields and divisor class groups is to perform index calculus directly on Abelian varieties. This variant is the subject of the following section.

15.8 Discrete Logarithms on Elliptic Curves over Extension Fields

We now discuss some related algorithms, which can be applied to elliptic curves over extension fields. We start by recalling Semaev's idea of summation polynomials.

15.8.1 Semaev's Summation Polynomials

Suppose that E is an elliptic curve defined over a prime field $\mathbb{F}_p$, and that elements of $\mathbb{F}_p$ are represented as integers in the interval $[0, p-1]$. Semaev [539] considered a factor base

$$\mathcal{B} = \{(x, y) \in E(\mathbb{F}_p) \ : \ 0 \leq x \leq p^{1/n}\}$$

for some fixed integer $n \geq 2$. Note that $\#\mathcal{B} \approx p^{1/n}$.

Semaev hoped to perform an index calculus algorithm similar to the one in Section 15.5.1. For random points $R = [a]P + [b]Q$ the task is to write R as a sum of points in $\mathcal{B}$. To accomplish this, Semaev introduced the notion of a summation polynomial.

Definition 15.8.1. Let $E : y^2 = x^3 + a_4x + a_6$ be an elliptic curve defined over $\mathbb{F}_q$, where the characteristic of $\mathbb{F}_q$ is neither 2 nor 3 (this condition can be avoided). The **summation polynomials** $\mathrm{Summ}_n \in \mathbb{F}_q[x_1, x_2, \ldots, x_n]$ for $n \geq 2$ are defined as follows:

- $\mathrm{Summ}_2(x_1, x_2) = x_1 - x_2$.
- $\mathrm{Summ}_3(x_1, x_2, x_3) = (x_1 - x_2)^2x_3^2 - 2((x_1 + x_2)(x_1x_2 + a_4) + 2a_6)x_3 + ((x_1x_2 - a_4)^2 - 4a_6(x_1 + x_2))$.
- $\mathrm{Summ}_n(x_1, x_2, \ldots, x_n) = R_x(\mathrm{Summ}_{n-1}(x_1, \ldots, x_{n-2}, x), \mathrm{Summ}_3(x_{n-1}, x_n, x))$ for $n \geq 4$ where $R_x(F, G)$ is the resultant of the polynomials F and G with respect to the variable x.

For many more details see Section 3 of [177]. The following result is from [539].

Theorem 15.8.2. *Summation polynomials have the following properties:*

- $(x_1, \ldots, x_n) \in \overline{\mathbb{F}}_q^n$ *is a root of* Summ_n *if and only if there exists* $(y_1, \ldots, y_n) \in \overline{\mathbb{F}}_q^n$ *such that* $P_i = (x_i, y_i) \in E(\overline{\mathbb{F}}_q)$ *and* $\sum_{i=1}^n P_i = \infty$.

- Summ_n *is symmetric.*
- *The degree of* Summ_n *in* x_i *is* 2^{n-2}.

Exercise 15.8.3. Prove Theorem 15.8.2.

One way to decompose $R = (x_R, y_R)$ in $\mathcal{B}$ is to find solutions $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ to

$$\text{Summ}_{n+1}(x_1, x_2, \ldots, x_n, x_R) \equiv 0 \pmod{p}, \text{ such that } 0 \leq x_i \leq p^{1/n}. \tag{15.12}$$

If such a solution exists and can be found then one finds the corresponding y-coordinates $\pm y_i$. Suppose that each $y_i \in \mathbb{F}_p$. Then each $P_i = (x_i, y_i)$ is in $\mathcal{B}$ and by Theorem 15.8.2 there exist $s_i \in \{-1, 1\}$ such that $s_1 P_1 + \cdots + s_n P_n = R$. The sign bits s_i can be found by exhaustive search, thereby yielding a relation. Since $\#\{P_1 + P_2 + \cdots + P_n : P_i \in \mathcal{B}\} \approx (p^{1/n})^n / n! = p/n!$ the expected number of points R that have to be selected before a relation is obtained is about $n!$.

Unfortunately, no efficient algorithm is known for solving the polynomial equation (15.12) even for $n = 5$ (in which case the equation has degree 16 in each of its 5 variables). Coppersmith's method (see Section 19.2) seems not to be useful for this task.

In reference to the remarks of Section 15.2.3 we see that all requirements for an index calculus algorithm are met, except that it is not efficient to decompose a smooth element over the factor base.

15.8.2 Gaudry's Variant of Semaev's Method

Gaudry [245] realised that it might be possible to take roots of summation polynomials if one was working with elliptic curves over extension fields. Gaudry's algorithm may be viewed as doing Weil descent without divisor class groups. Indeed, the paper [245] presents a general approach to index calculus on Abelian varieties and so the results apply in greater generality than just Weil descent of elliptic curves.

Suppose that E is an elliptic curve defined over a finite field $\mathbb{F}_{q^n}$ with $n > 1$. Gaudry [245] defines a factor base

$$\mathcal{B} = \{(x, y) \in E(\mathbb{F}_{q^n}) \ : \ x \in \mathbb{F}_q\}$$

so that $\#\mathcal{B} \approx q$. Gaudry considers this as the set of $\mathbb{F}_q$-rational points on the algebraic set F formed by intersecting the Weil restriction of scalars of E with respect to $\mathbb{F}_{q^n}/\mathbb{F}_q$ by $n-1$ hyperplanes $V(x_i)$ for $2 \leq i \leq n$, where $x = x_1\theta_1 + \cdots + x_n\theta_n$ (with $\theta_1 = 1$) as in Lemma 5.7.1. If the algebraic set F is irreducible then it is a 1-dimensional variety F.

In the relation generation stage, one attempts to decompose a randomly selected point $R \in E(\mathbb{F}_{q^n})$ as a sum of points in $\mathcal{B}$. Gaudry observed that this can be accomplished by finding solutions

$$(x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n \quad \text{such that} \quad \text{Summ}_{n+1}(x_1, x_2, \ldots, x_n, x_R) = 0. \tag{15.13}$$

Note that $\text{Summ}_{n+1}(x_1, \ldots, x_n, x_R) \in \mathbb{F}_{q^n}[x_1, \ldots, x_n]$ since E is defined over $\mathbb{F}_{q^n}$ and $x_R \in \mathbb{F}_{q^n}$. The conditions $x_j \in \mathbb{F}_q$ in equation (15.13) can be expressed algebraically as follows. Select a basis $\{\theta_1, \ldots, \theta_n\}$ for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ and write

$$\text{Summ}_{n+1}(x_1, \ldots, x_n, x_R) \ = \ \sum_{i=1}^{n} G_i(x_1, \ldots, x_n)\theta_i \tag{15.14}$$

where $G_i(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$. Note that the degree of G_i in x_j is at most 2^{n-1}. The polynomials G_i of equation (15.14) define an algebraic set in $X \subseteq \mathbb{A}^n$ and we are

interested in the points in $X(\mathbb{F}_q)$ (if there are any). Since $\mathbb{F}_q$ is finite there are only finitely many $\mathbb{F}_q$-rational solutions $(x_1, \ldots, x_n)$ to the system.

Gaudry assumes that X is generically a zero-dimensional algebraic set (Gaudry justifies this assumption by noting that if F is a variety then the variety F^n is n-dimensional, and so the map from F^n to the Weil restriction of E, given by adding together n points in F, is a morphism between varieties of the same dimension, and so generically has finite degree). The $\mathbb{F}_q$-rational solutions can therefore be found by finding a Gröbner basis for the ideal generated by the G_i and then taking roots in $\mathbb{F}_q$ of a sequence of univariate polynomials each of which has degree at most $2^{n(n-1)}$. This is predicted to take $O(2^{cn(n-1)}M(\log(q)))$ bit operations for some constant c. Alternatively one could add some field equations $x_j^q - x_j$ to the ideal, to ensure it is zero-dimensional, but this could have an adverse effect on the complexity. Gaudry makes a further heuristic assumption, namely that the smoothness probability behaves as expected when using the large prime variant.

The size of the set $\{P_1 + P_2 + \cdots + P_n : P_i \in \mathcal{B}\}$ is approximately $q^n/n!$ and so the expected number of points R that have to be selected before a relation is obtained is about $n!$. One needs approximately $\#\mathcal{B} \approx q$ relations to be able to find a non-trivial element in the kernel of the relation matrix and hence integers a and b such that $[a]D_1 + [b]D_2 \equiv 0$. It follows that the heuristic expected running time of Gaudry's algorithm is

$$\tilde{O}(2^{cn(n-1)}n!qM(\log(q)) + q^{2+o(1)}) \tag{15.15}$$

bit operations as $q \to \infty$. This is exponential in terms of n and $\log(q)$. However, for fixed n, the running time can be expressed as $\tilde{O}(q^2)$ bit operations.

Gaudry's focus was on n fixed and relatively small. For any fixed $n \geq 5$, Gaudry's heuristic algorithm for solving the ECDLP over $\mathbb{F}_{q^n}$ is asymptotically faster than Pollard's rho method. The double large prime variant (mentioned in Section 15.6.3) can also be used in this setting. The complexity therefore becomes (heuristic) $\tilde{O}(q^{2-\frac{2}{n}})$ bit operations. Hence Gaudry's algorithm is asymptotically faster than Pollard rho even for $n = 3$ and $n = 4$, namely $\tilde{O}(q^{4/3})$ rather than $\tilde{O}(q^{3/2})$ for $n = 3$ and $\tilde{O}(q^{3/2})$ rather than $\tilde{O}(q^2)$ for $n = 4$.

15.8.3 Diem's Algorithm for the ECDLP

Gaudry's focus was on the DLP in $E(\mathbb{F}_{q^n})$ when n is fixed. This yields an exponential-time algorithm. Diem [173, 177] considered the case where n is allowed to grow, and obtained a subexponential-time algorithm.

The crux of Diem's method is remarkably simple: he assumes $n \approx \sqrt{\log(q)}$ and obtains an algorithm for the DLP in $E(\mathbb{F}_{q^n})$ with complexity $O(q^c)$ for some constant c (note that even some exponential-time computations in n are polynomial in q as $e^{n^2} \approx q$). Now, $q^c = \exp(c\log(q))$ and $\log(q^n) = n\log(q) \approx \log(q)^{3/2}$ so $q^c \approx \exp(c\log(q^n)^{2/3}) < L_{q^n}(2/3, c)$.

Diem's algorithm is very similar to Gaudry's. In Gaudry's algorithm, the factor base consists of points whose x-coordinates lie in $\mathbb{F}_q$. Diem defines a function $\varphi = \alpha \circ x$, where α is an automorphism over $\mathbb{F}_{q^n}$ of $\mathbb{P}^1$ that satisfies a certain condition, and defines the factor base to be $\mathcal{B} = \{P \in E(\mathbb{F}_{q^n}) : \varphi(P) \in \mathbb{P}^1(\mathbb{F}_q)\}$. The process of generating relations proceeds in the standard way. Some important contributions of [177] are to prove that the algebraic set defined by the summation polynomials has a good chance of having dimension zero, and that when this is the case the points can be found by taking resultants of multihomogeneous polynomials in time polynomial in $e^{n^2}\log(q)$ (which is exponential in n but polynomial in q).

The main result of [177] is the following. We stress that this result does not rely on any heuristics.

Theorem 15.8.4. *(Diem) Let $a, b \in \mathbb{R}$ be such that $0 < a < b$. There is an algorithm such that, if q is a prime power and $n \in \mathbb{N}$ is such that*

$$a\sqrt{\log(q)} \leq n \leq b\sqrt{\log(q)}$$

and E is any elliptic curve over $\mathbb{F}_{q^n}$, then the algorithm solves the DLP in $E(\mathbb{F}_{q^n})$ in an expected $e^{O(\log(q^n)^{2/3})}$ bit operations.

15.9 Further Results

To end the chapter we briefly mention some methods for non-hyperelliptic curves. It is beyond the scope of the book to present these algorithms in detail. We then briefly summarise the argument that there is no subexponential algorithm for the DLP on elliptic curves in general.

15.9.1 Diem's Algorithm for Plane Curves of Low Degree

Diem [175] used the Adleman-DeMarrais-Huang idea of generating relations using principal divisors $a(x) - yb(x)$ for the DLP on plane curves $F(x, y) = 0$ of low degree (the degree of such a curve is the total degree of $F(x, y)$ as a polynomial). Such curves are essentially the opposite case to hyperelliptic curves (which have rather high degree in x relative to their genus). The trick is simply to note that if $F(x, y)$ has relatively low degree compared to its genus then so does $b(x)^d F(x, a(x))$ and so the divisor of the function $a(x) - yb(x)$ has relatively low weight. The main result is an algorithm with heuristic complexity $\tilde{O}(q^{2-2/(d-2)})$ bit operations for a curve of degree d over $\mathbb{F}_q$.

In the case of non-singular plane quartics (genus 3 curves C over $\mathbb{F}_q$) Diem takes the factor base to be a large set of points $\mathcal{B} \subseteq C(\mathbb{F}_q)$. He generates relations by choosing two distinct points $P_1, P_2 \in \mathcal{B}$ and intersecting the line $y = bx + c$ between them with the curve C. There are two other points of intersection, corresponding to the roots of the quadratic polynomial $F(x, bx + c)/((x - x_{P_1})(x - x_{P_2}))$ and so with probability roughly $1/2$ we expect to get a relation in the divisor class group among points in $C(\mathbb{F}_q)$. Diem shows that the algorithm has complexity $\tilde{O}(q)$ bit operations.

Due to lack of space, and since our focus in this book is hyperelliptic curves (though, it is important to note that Smith [573] has given a reduction of the DLP from hyperelliptic curves of genus 3 to plane quartics) we do not present any further details. Interested readers should see [175, 178].

15.9.2 The Algorithm of Enge-Gaudry-Thomé and Diem

The algorithms for the DLP in the divisor class group of a hyperelliptic curve in Sections 15.6.1 and 15.6.2 had complexity $L_{q^g}(1/2, \sqrt{2} + o(1))$ bit operations as $q \to \infty$. A natural problem is to find algorithms with complexity $L_{q^g}(1/3, c + o(1))$, and this is still open in general. However, an algorithm is known for curves of the form $y^n + F(x, y) = 0$ where $\deg_y(F(x, y)) \leq n - 1$ and $\deg_x(F(x, y)) = d$ for $n \approx g^{1/3}$ and $d \approx g^{2/3}$. We do not have space to give the details, so simply quote the results and refer to Enge and Gaudry [196], Enge, Gaudry and Thomé [197] and Diem [174]. An algorithm to compute the group structure of $\mathrm{Pic}^0_C(\mathbb{F}_q)$ is given with heuristic complexity of $L_{q^g}(1/3, c + o(1))$

bit operations for some constant c. For the discrete logarithm problem the algorithm has heuristic complexity $L_{q^g}(1/3, c' + o(1))$ bit operations where c' is a constant.

Unlike the $L_N(1/3, c + o(1))$ algorithms for factoring or DLP in finite fields, the algorithm does not use two different factor bases. Instead, the algorithm is basically the same idea as Sections 15.6.2 and 15.9.1 with a complexity analysis tailored for curves of a certain form.

15.9.3 Index Calculus for General Elliptic Curves

In this section we briefly discuss why there does not seem to be a subexponential algorithm for the DLP on general elliptic curves.

An approach to an index calculus algorithm for elliptic curves was already discussed by Miller [428] in the paper that first proposed elliptic curves for cryptography. In particular he considered "lifting" an elliptic curve E over $\mathbb{F}_p$ to an elliptic curve $\widetilde{E}$ over $\mathbb{Q}$ (i.e., so that reducing the coefficients of $\widetilde{E}$ modulo p yields E). The factor base $\mathcal{B}$ was defined to be the points of small height (see Section VIII.6 of [564] for details of heights) in $\widetilde{E}(\mathbb{Q})$. The theory of descent (see Chapter VIII of Silverman [564]) essentially gives an algorithm to decompose a point as a sum of points of small height (when this is possible). The idea would therefore be to take random points $[a]P + [b]Q \in E(\mathbb{F}_p)$, lift them to $\widetilde{E}(\mathbb{Q})$ and then decompose them over the factor base. There are several obstructions to this method. First, lifting a random point from $E(\mathbb{F}_p)$ to $\widetilde{E}(\mathbb{Q})$ seems to be hard in general. Indeed, Miller argued (see also [566]) that there are very few points of small height in $\widetilde{E}(\mathbb{Q})$ and so (since we are considering random points $[a]P + [b]Q$ from the exponentially large set $E(\mathbb{F}_p)$) it would be necessary to lift to exponentially large points in $\widetilde{E}(\mathbb{Q})$. Second, the lifting itself seems to be a non-trivial computational task (essentially, solving a non-linear Diophantine equation over $\mathbb{Z}$).

Silverman proposed the **Xedni calculus** attack[11], which was designed to solve the lifting problem. This algorithm was analysed in [322], where it is shown that the probability of finding useful relations is too low.

By now, many people have tried and failed to discover an index calculus algorithm for the DLP on general elliptic curves. However, this does not prove that no such algorithm exists, or that a different paradigm could not lead to faster attacks on the elliptic curve DLP.

[11] "Xedni" is "Index" spelled backwards.