Chapter 12

Primality Testing and Integer Factorisation using Algebraic Groups

There are numerous books about primality testing and integer factorisation, of which the most notable is Crandall and Pomerance [162]. There is no need to reproduce all the details of these topics. Hence, the purpose of this chapter is simply to sketch a few basic ideas that will be used later. In particular, we describe methods for primality testing and integer factorisation that exploit the structure of algebraic groups.

Definition 12.0.1. A **primality test** is a randomised algorithm that, on input $N \in \mathbb{N}$, outputs a single bit b such that if N is prime then $b = 1$. A composite integer that passes a primality test is called a **pseudoprime**. An algorithm **splits** $N \in \mathbb{N}$ if it outputs a pair (a, b) of integers such that $1 < a, b < N$ and $N = ab$.

12.1 Primality Testing

The simplest primality test is **trial division** (namely, testing whether N is divisible by any integer up to $\sqrt{N}$). This algorithm is not useful for factoring numbers chosen for cryptography, but the first step of most general purpose factoring algorithms is to run trial division to remove all 'small' prime factors of N before trying more elaborate methods. Hence, for the remainder of this section we may assume that N is odd (and usually that it is not divisible by any primes less than, say, 10^6).

12.1.1 Fermat Test

Let $N \in \mathbb{N}$. If N is prime then the algebraic group $G_m(\mathbb{Z}/N\mathbb{Z}) = (\mathbb{Z}/N\mathbb{Z})^*$ over the ring $\mathbb{Z}/N\mathbb{Z}$ has $N-1$ elements. In other words, if a is an integer such that $\gcd(a, N) = 1$ and

$$a^{N-1} \not\equiv 1 \pmod{N}$$

then N is not prime. Such a number a is called a **compositeness witness** for N. The hope is that if N is not prime then the order of the group $G_m(\mathbb{Z}/N\mathbb{Z})$ is not a divisor of $N-1$ and so a compositeness witness exists. Hence, the Fermat test is to choose random $1 < a < N$ and compute $a^{N-1} \pmod{N}$.

As is well-known, there are composite numbers N that are pseudoprimes for the Fermat test.

Definition 12.1.1. An integer $N \in \mathbb{N}$ is a **Carmichael number** if N is composite and

$$a^{N-1} \equiv 1 \pmod{N}$$

for all $a \in \mathbb{N}$ such that $\gcd(a, N) = 1$.

If $N = \prod_{i=1}^{l} p_i^{e_i}$ is composite then $G_m(\mathbb{Z}/N\mathbb{Z}) \cong \prod_{i=1}^{l} G_m(\mathbb{Z}/p_i^{e_i}\mathbb{Z})$ and has order $\varphi(N)$ and exponent $\lambda(N) = \operatorname{lcm}\{p_i^{e_i-1}(p_i - 1) : 1 \leq i \leq l\}$.

Exercise 12.1.2. Show that all Carmichael numbers are odd. Show that N is a Carmichael number if and only if $\lambda(N) \mid (N-1)$. Show that a composite number $N \in \mathbb{N}$ is a Carmichael number if and only if $N = \prod_{i=1}^{l} p_i$ is a product of distinct primes such that $(p_i - 1) \mid (N-1)$ for $i = 1, \ldots, l$.

Exercise 12.1.3. Show that $561 = 3 \cdot 11 \cdot 17$ is a Carmichael number.

It was shown by Alford, Granville and Pomerance [10] in 1992 that there are infinitely many Carmichael numbers.

It is natural to replace $G_m(\mathbb{Z}/N\mathbb{Z})$ with any algebraic group or algebraic group quotient, such as the torus $\mathbb{T}_2$, the algebraic group quotient corresponding to Lucas sequences (this gives rise to the $p+1$ test) or an elliptic curve of predictable group order.

Exercise 12.1.4. Design a primality test based on the algebraic group $\mathbb{T}_2(\mathbb{Z}/N\mathbb{Z})$, which has order $N+1$ if N is prime. Also show to use Lucas sequences to test N for primality using the algebraic group quotient.

Exercise 12.1.5. Design a primality test for integers $N \equiv 3 \pmod{4}$ based on the algebraic group $E(\mathbb{Z}/N\mathbb{Z})$ where E is a suitably chosen supersingular elliptic curve.

Exercise 12.1.6. Design a primality test for integers $N \equiv 1 \pmod{4}$ based on the algebraic group $E(\mathbb{Z}/N\mathbb{Z})$ where E is a suitably chosen elliptic curve.

12.1.2 The Miller-Rabin Test

This primality test is also called the Selfridge-Miller-Rabin test or the strong prime test. It is a refinement of the Fermat test, and works very well in practice. Rather than changing the algebraic group, the idea is to make better use of the available information. It is based on the following trivial lemma, which is false if p is replaced by a composite number N (except for $N = p^a$ where p is odd).

Lemma 12.1.7. *Let p be prime. If $x^2 \equiv 1 \pmod{p}$ then $x \equiv \pm 1 \pmod{p}$.*

For the Miller-Rabin test write $N-1 = 2^b m$ where m is odd and consider the sequence $a_0 = a^m \pmod N$, $a_1 = a_0^2 = a^{2m} \pmod N$, $\ldots$, $a_b = a_{b-1}^2 = a^{N-1} \pmod N$ where $\gcd(a, N) = 1$. If N is prime then this sequence must have the form $(*, *, \ldots, *, -1, 1, \ldots, 1)$ or $(-1, 1, \ldots, 1)$ or $(1, \ldots, 1)$ (where $*$ denotes numbers whose values are not relevant). Any deviation from this form means that the number N is composite.

An integer N is called a **base-a probable prime** if the Miller-Rabin sequence has the good form and is called a **base-a pseudoprime** if it is a base-a probable prime that is actually composite.

Exercise 12.1.8. Let $N = 561$. Note that $\gcd(2, N) = 1$ and $2^{N-1} \equiv 1 \pmod N$. Show that the Miller-Rabin method with $a = 2$ demonstrates that N is composite. Show that this failure allows one to immediately split N.

Theorem 12.1.9. *Let $n > 9$ be an odd composite integer. Then N is a base-a pseudoprime for at most $\varphi(N)/4$ bases between 1 and N.*

Proof: See Theorem 3.5.4 of [162] or Theorem 10.6 of Shoup [556]. □

Hence, if a number N passes several Miller-Rabin tests for several randomly chosen bases a then one can believe that with high probability N is prime (Section 5.4.2 of Stinson [592] gives a careful analysis of the probability of success of a closely related algorithm using Bayes' theorem). Such an integer is called a **probable prime**. In practice one chooses $O(\log(N))$ random bases a and runs the Miller-Rabin test for each. The total complexity is therefore $O(\log(N)^4)$ bit operations (which can be improved to $O(\log(N)^2 M(\log(N)))$, where $M(m)$ is the cost of multiplying two m-bit integers).

12.1.3 Primality Proving

Agrawal, Kayal and Saxena [6] (AKS) discovered a deterministic algorithm that runs in polynomial-time and determines whether or not N is prime. We refer to Section 4.5 of [162] for details. The original AKS test has been improved significantly. A variant due to Bernstein requires $O(\log(N)^{4+o(1)})$ bit operations using fast arithmetic (see Section 4.5.4 of [162]).

There is also a large literature on primality proving using Gauss and Jacobi sums, and using elliptic curves. We refer to Sections 4.4 and 7.6 of [162].

In practice the Miller-Rabin test is still widely used for cryptographic applications.

12.2 Generating Random Primes

Definition 12.2.1. Let $X \in \mathbb{N}$, then $\pi(X)$ is defined to be the number of primes $1 < p < X$.

The famous **prime number theorem** states that $\pi(X)$ is asymptotically equal to $X/\log(X)$ (as always log denotes the natural logarithm). In other words, primes are rather common among the integers. If one choose a random integer $1 < p < X$ then the probability that p is prime is therefore about $1/\log(X)$ (equivalently, about $\log(X)$ trials are required to find a prime between 1 and X). In practice, this probability increases significantly if one choose p to be odd and not divisible by 3.

Theorem 12.2.2. *Random (probable) prime numbers of a given size X can be generated using the Miller-Rabin algorithm in expected $O(\log(X)^5)$ bit operations (or $O(\log(X)^3 M(\log(X)))$ using fast arithmetic).*

Exercise 12.2.3. For certain cryptosystems based on the discrete logarithm problem it is required to produce a k_1-bit prime p such that $p-1$ has a k_2-bit prime factor q. Give a method that takes integers k_1, k_2 such that $k_2 < k_1$ and outputs p and q such that p is a k_1-bit prime, q is a k_2-bit prime and $q \mid (p-1)$.

Exercise 12.2.4. For certain cryptosystems based on the discrete logarithm problem (see Chapter 6) it is required to produce a k_1-bit prime p such that $\Phi_k(p)$ has a k_2-bit prime factor q (where $\Phi_k(x)$ is the k-th cyclotomic polynomial). Give a method that takes integers k, k_1, k_2 such that $k_2 < \varphi(k)k_1$ and outputs p and q such that p is a k_1-bit prime, q is a k_2-bit prime and $q \mid \Phi_k(p)$.

Exercise 12.2.5. A **strong prime** is defined to be a prime p such that $q = (p-1)/2$ is prime, $(p+1)/2$ is prime and $(q-1)/2$ is prime (it is conjectured that infinitely many such primes exist). Some RSA systems require the RSA moduli to be a product of strong primes. Give an algorithm to generate strong primes.

12.2.1 Primality Certificates

For cryptographic applications it may be required to provide a **primality certificate**. This is a mathematical proof that can be checked in polynomial-time and that establishes the primality of a number n. Pratt [490] showed that there exists a short primality certificate for every prime. Primality certificates are not so important since the discovery of the AKS test, but primes together with certificates can be generated (and the certificates verified) more quickly than using the AKS test, so this subject could still be of interest. We refer to Section 4.1.3 of Crandall and Pomerance [162] and Maurer [403] for further details.

One basic tool for primality certificates is Lucas' converse of Fermat's little theorem.

Theorem 12.2.6. *(Lucas) Let $N \in \mathbb{N}$. If there is an integer a such that $\gcd(a, N) = 1$, $a^{N-1} \equiv 1 \pmod{N}$ and $a^{(N-1)/l} \not\equiv 1 \pmod{N}$ for all primes $l \mid (N-1)$ then N is prime.*

Exercise 12.2.7. Prove Theorem 12.2.6.

In practice one can weaken the hypothesis of Theorem 12.2.6.

Theorem 12.2.8. *(Pocklington) Suppose $N - 1 = FR$ where the complete factorisation of F is known. Suppose there is an integer a such that $a^{N-1} \equiv 1 \pmod{N}$ and*

$$a^{(N-1)/q} \not\equiv 1 \pmod{N}$$

for every prime $q \mid F$. Then every prime factor of N is congruent to 1 modulo F. Hence, if $F \geq \sqrt{N}$ then N is prime.

Exercise 12.2.9. Prove Theorem 12.2.8.

Exercise 12.2.10. A **Sophie-Germain prime** (in cryptography the name **safe prime** is commonly used) is a prime p such that $(p-1)/2$ is also prime. It is conjectured that there are infinitely many Sophie-Germain primes. Give a method to generate a k-bit Sophie-Germain prime together with a certificate of primality, such that the output is close to uniform over the set of all k-bit Sophie-Germain primes.

12.3 The $p-1$ Factoring Method

First we recall the notion of a smooth integer. These are discussed in more detail in Section 15.1.

Definition 12.3.1. Let $N = \prod_{i=1}^{r} p_i^{e_i} \in \mathbb{N}$ (where we assume the p_i are distinct primes and $e_i \geq 1$) and let $B \in \mathbb{N}$. Then N is **B-smooth** if all $p_i \leq B$ and N is **B-power smooth** (or **strongly B-smooth**) if all $p_i^{e_i} \leq B$.

Example 12.3.2. $528 = 2^4 \cdot 3 \cdot 11$ is 14-smooth but is not 14-power smooth.

The $p-1$ method was published by Pollard [485].[1] The idea is to suppose that N has prime factors p and q where $p-1$ is B-power smooth but $q-1$ is not B-power smooth. Then if $1 < a < N$ is randomly chosen we have $a^{B!} \equiv 1 \pmod{p}$ and, with high probability, $a^{B!} \not\equiv 1 \pmod{q}$. Hence $\gcd(a^{B!} - 1, N)$ splits N. Algorithm 11 gives the Pollard $p-1$ algorithm.

Example 12.3.3. Let $N = 124639$ and let $B = 8$. Choose $a = 2$. One can check that

$$\gcd(a^{B!} \pmod{N} - 1, N) = 113$$

from which one deduces that $N = 113 \cdot 1103$.

This example worked because the prime $p = 113$ satisfies $p - 1 = 2^4 \cdot 7 \mid 8!$ and so $2^{8!} \equiv 1 \pmod{p}$ while the other prime satisfies $q - 1 = 2 \cdot 19 \cdot 29$, which is not 8-smooth.

Of course, the "factor" returned from the gcd may be 1 or N. If the factor is not 1 or N then we have split N as $N = ab$. We now test each factor for primality and attempt to split any composite factors further.

Algorithm 11 Pollard $p-1$ algorithm

INPUT: $N \in \mathbb{N}$
OUTPUT: Factor of N
1: Choose a suitable value for B
2: Choose a random $1 < a < N$
3: $b = a$
4: **for** $i = 2$ to B **do**
5: $\quad b = b^i \pmod{N}$
6: **end for**
7: **return** $\gcd(b - 1, N)$

Exercise 12.3.4. Factor $N = 10028219737$ using the $p-1$ method.

Lemma 12.3.5. *The complexity of Algorithm 11 is $O(B \log(B) M(\log(N)))$ bit operations.*

Proof: The main loop is repeated B times and contains an exponentiation modulo N to a power $i < B$. The cost of the exponentiation is $O(\log(B)M(\log(N)))$ bit operations. □

The algorithm is therefore exponential in B and so is only practical if B is relatively small. If $B = O(\log(N)^i)$ then the algorithm is polynomial-time. Unfortunately, the algorithm only splits numbers of a special form (namely those for which there is a factor p such that $p-1$ is very smooth).

[1] According to [634] the first stage of the method was also known to D. N. and D. H. Lehmer, though they never published it.

Exercise 12.3.6. Show that searching only over prime power values for i in Algorithm 11 lowers the complexity to $O(BM(\log(N)))$ bit operations.

It is usual to have a **second stage** or **continuation** to the Pollard $p-1$ method. Suppose that Algorithm 11 terminates with $\gcd(b-1, N) = 1$. If there is a prime $p \mid N$ such that $p - 1 = SQ$ where S is B-smooth and Q is prime then the order of b modulo p is Q. One will therefore expect to split N by computing $\gcd(b^Q \pmod N - 1, N)$. The second stage is to find Q if it is not too big. One therefore chooses a bound $B' > B$ and wants to compute $\gcd(b^Q \pmod N - 1, N)$ for all primes $B < Q \leq B'$.

We give two methods to do this: the standard continuation (Exercise 12.3.7) has the same complexity as the first stage of the $p-1$ method, but the constants are much better; the FFT continuation (Exercise 12.3.8) has better complexity and shows that if sufficient storage is available then one can take B' to be considerably bigger than B. Further improvements are given in Sections 4.1 and 4.2 of Montgomery [436].

Exercise 12.3.7. (Standard continuation) Show that one can compute $\gcd(b^Q \pmod N - 1, N)$ for all primes $B < Q \leq B'$ in $O((B' - B)M(\log(N)))$ bit operations.

Exercise 12.3.8. (Pollard's FFT continuation) Let $w = \lceil \sqrt{B' - B} \rceil$. We will exploit the fact that $Q = B + vw - u$ for some $0 \leq u < w$ and some $1 \leq v \leq w$ (this is very similar to the baby-step-giant-step algorithm; see Section 13.3). Let $P(x) = \prod_{i=0}^{w-1}(x - b^i) \pmod N$, computed as in Section 2.16. Now compute $\gcd(P(g^{B+vw}) \pmod N, N)$ for $v = 1, 2, \ldots, w$. For the correct value v we have

$$\begin{aligned} P(g^{B+vw}) = \prod_i (g^{B+vw} - g^i) &= (g^{B+vw} - g^u) \prod_{i \neq u} (g^{B+vw} - g^i) \\ &= g^u (g^{B+vw-u} - 1) \prod_{i \neq u} (g^{B+vw} - g^i). \end{aligned}$$

Since $g^{B+vw-u} = g^Q \equiv 1 \pmod p$ then $\gcd(P(g^{B+vw}) \pmod N, N)$ is divisible by p. Show that the time complexity of this continutation is $O(M(w)\log(w)M(\log(N))$, which asymptotically is $O(\sqrt{B'}\log(B')^2 \log(\log(B'))M(\log(N)))$, bit operations. Show that the storage required is $O(w\log(w)) = O(\sqrt{B'}\log(B'))$ bits.

Exercise 12.3.9. The $p+1$ factoring method uses the same idea as the $p-1$ method, but in the algebraic group $\mathbb{T}_2$ or the algebraic group quotient corresponding to Lucas sequences. Write down the details of the $p+1$ factoring method using Lucas sequences.

12.4 Elliptic Curve Method

Let N be an integer to be factored and let $p \mid N$ be prime. One can view Pollard's $p-1$ method as using an auxiliary group (namely, $G_m(\mathbb{F}_p)$) that may have smooth order. The idea is then to obtain an element modulo N (namely, $a^{B!}$) that is congruent modulo p (but not modulo some other prime $q \mid N$) to the identity element of the auxiliary group.

Lenstra's idea was to replace the group G_m in the Pollard $p-1$ method with the group of points on an elliptic curve. The motivation was that even if $p-1$ is not smooth, it is reasonable to expect that there is an elliptic curve E over $\mathbb{F}_p$ such that $\#E(\mathbb{F}_p)$ is rather smooth. Furthermore, since there are lots of different elliptic curves over the field $\mathbb{F}_p$ we have a chance to split N by trying the method with lots of different elliptic curves. We refer to Section 9.14 for some remarks on elliptic curves modulo N.

If E is a "randomly chosen" elliptic curve modulo N with a point P on E modulo N then one hopes that the point $Q = [B!]P$ is congruent modulo p (but not modulo

some other prime q) to the identity element. One constructs E and P together, for example choosing $1 < x_P, y_P, a_4 < N$ and setting $a_6 = y_P^2 - x_P^3 - a_4 x_P \pmod N$. If one computes $Q = (x : y : z)$ using inversion-free arithmetic and projective coordinates (as in Exercise 9.1.5) then $Q \equiv \mathcal{O}_E \pmod p$ is equivalent to $p \mid z$. Here we are performing elliptic curve arithmetic over the ring $\mathbb{Z}/N\mathbb{Z}$ (see Section 9.14).

The resulting algorithm is known as the **elliptic curve method** or **ECM** and it is very widely used, both as a general-purpose factoring algorithm in computer algebra packages, and as a subroutine of the number field sieve. An important consequence of Lenstra's suggestion of replacing the group $\mathbb{F}_p^*$ by $E(\mathbb{F}_p)$ is that it motivated Miller and Koblitz to suggest using $E(\mathbb{F}_p)$ instead of $\mathbb{F}_p^*$ for public key cryptography.

Algorithm 12 gives a sketch of one round of the ECM algorithm. If the algorithm fails then one should repeat it, possibly increasing the size of B. Note that it can be more efficient to compute $[B!]P$ as a single exponentiation rather than a loop as in line 5 of Algorithm 12; see [49].

Algorithm 12 Elliptic curve factoring algorithm

INPUT: $N \in \mathbb{N}$
OUTPUT: Factor of N
1: Choose a suitable value for B
2: Choose random elements $0 \leq x, y, a_4 < N$
3: Set $a_6 = y^2 - x^3 - a_4 x \pmod N$
4: Set $P = (x : y : 1)$
5: **for** $i = 2$ to B **do**
6: Compute $P = [i]P$
7: **end for**
8: **return** $\gcd(N, z)$ where $P = (x : y : z)$

Exercise 12.4.1. Show that the complexity of Algorithm 12 is $O(B \log(B) M(\log(N)))$ bit operations.

Exercise 12.4.2. Show that the complexity of Algorithm 12 can be lowered to $O(BM(\log(N)))$ bit operations using the method of Exercise 12.3.6.

Many of the techniques used to improve the Pollard $p-1$ method (such as the standard continuation, though not Pollard's FFT continuation) also apply directly to the elliptic curve method. We refer to Section 7.4 of [162] for details. One can also employ all known techniques to speed up elliptic curve arithmetic. Indeed, the Montgomery model for elliptic curves (Section 9.12.1) was discovered in the context of ECM rather than ECC.

In practice, we repeat the algorithm a number of times for random choices of B, x, y and a_4. The difficult problems are to determine a good choice for B and to analyse the probability of success. We discuss these issues in Section 15.3 where we state Lenstra's conjecture that the elliptic curve method factors integers in subexponential time.

12.5 Pollard-Strassen Method

Pollard [485] and, independently, Strassen gave a deterministic algorithm to factor an integer N in $\tilde{O}(N^{1/4})$ bit operations. It is based on the idea[2] of Section 2.16, namely that

[2]Despite the title of this chapter, the Pollard-Strassen algorithm does not use algebraic groups, or any group-theoretic property of the integers modulo N.

one can evaluate a polynomial of degree n in $(\mathbb{Z}/N\mathbb{Z})[x]$ at n values in $O(M(n)\log(n))$ operations in $\mathbb{Z}/N\mathbb{Z}$. A different factoring algorithm with this complexity is given in Exercise 19.4.7.

The trick to let $B = \lceil N^{1/4} \rceil$, $F(x) = x(x-1)\cdots(x-B+1)$ (which has degree B) and to compute $F(jB) \pmod N$ for $1 \leq j \leq B$. Computing these values requires $O(M(B)\log(B)M(\log(N))) = O(N^{1/4}\log(N)^3\log(\log(N))^2\log(\log(\log(N))))$ bit operations. Once this list of values has been computed one computes $\gcd(N, F(jB) \pmod N)$ until one finds a value that is not 1. This will happen, for some j, since the smallest prime factor of N is of the form $jB - i$ for some $1 \leq j \leq B$ and some $0 \leq i < B$. Note that $M = \gcd(N, F(jB) \pmod N)$ may not be prime, but one can find the prime factors of it in $\tilde{O}(N^{1/4})$ bit operations by computing $\gcd(M, jB - i)$ for that value of j and all $0 \leq i < B$. Indeed, one can find all prime factors of N that are less than $N^{1/2}$ (and hence factor N completely) using this method. The overall complexity is $\tilde{O}(N^{1/4})$ bit operations.

Exercise 12.5.1.★ Show that one can determine all primes p such that $p^2 \mid N$ in $\tilde{O}(N^{1/6})$ bit operations.