

## Chapter 2

# Applications: Algorithms, Primality and Factorization, Codes

*“Elle est retrouvée.  
Quoi ? - L’Éternité.  
C’est la mer allée  
Avec le soleil.”*

ARTHUR RIMBAUD

*This chapter describes some industrial applications of number theory, via computer science. We succinctly describe the main algorithms as well as their theoretical complexity or computation time. We use the notation  $O(f(n))$  to denote a function  $\leq Cf(n)$ ; furthermore, the unimportant—at least from a theoretical point of view—constants which appear will be ignored. In the following sections, we introduce the basics of cryptography and of the “RSA” system, which motivates the study of primality tests and factorization methods. We finish the chapter with an introduction to error-correcting codes, which will lead us into the study of cyclotomic polynomials.*

### 1. Basic Algorithms

Let  $n$  be an integer. Once we have chosen a base  $b \geq 2$ , we write  $n$  in base  $b$ , in other words, with the *digits*  $a_i \in [0, b - 1]$ :

$$n = a_0 + a_1b + \cdots + a_rb^r = \overline{a_ra_{r-1} \dots a_1a_0}^b, \text{ where } a_r \neq 0$$

(the two most standard base choices are  $b = 10$  for usual decimal notation and  $b = 2$  for binary notation, which is especially well-adapted to computer

programming). We will consider an operation on the digits to be a single operation (or an operation which needs  $O(1)$  computation time). It is natural to refer to the number of digits necessary in order to describe  $n$ , in other words  $r + 1$ , as its *complexity*. Since we can see that  $b^r \leq a_r b^r < n \leq b^{r+1}$ , we know that

$$r \leq \frac{\log n}{\log b} < r + 1$$

and can therefore describe the complexity as proportional to  $\log n$ . It is clear that the manipulation of random numbers of size  $n$  requires at least  $\log n$  elementary operations. We consider, as much from a practical point of view as from a theoretical one, an algorithm to be “good” if it is a *polynomial* algorithm; that is to say, it uses  $O((\log n)^\kappa)$  elementary operations. Conversely, we consider an *exponential* algorithm, meaning that its execution time or required number of operations is greater than  $\exp(\kappa \log n) = n^\kappa$ , to be infeasible (for large  $n$ , of course).

**Addition.** In order to add two numbers  $m$  and  $n$  with at most  $r$  digits, we must perform at most  $r$  additions of two digits and (possibly) carry a digit. The cost is therefore  $O(\log \max(n, m)) = O(r)$ . The number of operations used in subtraction is similar.

**Multiplication.** In order to calculate  $n \times m$ , where  $n$  and  $m$  are two numbers with at most  $r$  digits (with the usual elementary school algorithm), we must perform at most  $r^2$  elementary multiplications and  $r$  additions, and possibly carry a digit, and therefore, the cost is  $O((\log \max(n, m))^2) = O(r^2)$ .

**Remark.** The addition algorithm is (up to constants) optimal, but some more sophisticated methods (notably the “fast Fourier transform”) lets us perform multiplications at a much better cost, for example in  $O(r(\log r)^2)$ . See Exercises 2-7.3 and 2-7.4.

**Division algorithm.** Given  $a$  and  $b \geq 1$ , if we compute  $(q, r)$  such that  $a = qb + r$  and  $0 \leq r \leq b - 1$  with (a variation of) the algorithm learned in elementary school, we perform a number of elementary operations similar to that of multiplication, i.e.,  $O(\log \max(a, b)^2)$ . In order to give an example of a *turtle algorithm* (do not use!), we could perform the following procedure. We start by setting  $q_0 = 0$  and  $r_0 = a$ . Then we have  $a = q_0 b + r_0$ ; if  $r_0 < b$ , we stop, and if not, we compute  $q_1 = q_0 + 1$  and  $r_1 = r_0 - b$  in such a way that  $a = q_1 b + r_1$ , and we get the result by iteration and by stopping when  $r_n < b$  and  $a = q_n b + r_n$ . If  $a > b$ , we must perform approximately  $a/b$  subtractions, therefore the cost is  $O((\log a) \times (a/b))$  (which is exponential).

**Euclidean algorithm.** Given two integers,  $a$  and  $b$ , the goal is to compute  $d := \gcd(a, b)$  and  $(u, v) \in \mathbf{Z}^2$  such  $au + bv = d$  (Bézout’s lemma). The

principle is the following: we divide  $a$  by  $b$ ,  $a = bq_1 + r_1$ ; then divide  $b$  by  $r_1$ ,  $b = r_1q_2 + r_2$ , and in subsequent steps divide  $r_n$  by  $r_{n+1}$ ,  $r_n = r_{n+1}q_{n+2} + r_{n+2}$ . Keep in mind that the sequence  $r_n$  is strictly decreasing and stops when  $r_{n+1} = 0$ , and therefore  $\gcd(a, b) = r_n$ . In fact,

$$\gcd(a, b) = \gcd(b, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_n, r_{n+1}) = r_n.$$

In order to compute  $(u, v)$ , we could proceed as follows: we set  $u_0 = 1$ ,  $u_1 = 0$ ,  $v_0 = 0$  and  $v_1 = 1$  and then recursively define  $u_n = u_{n-2} - q_n u_{n-1}$  and  $v_n = v_{n-2} - q_n v_{n-1}$ . One can immediately check by induction that  $au_n + bv_n = r_n$ . We will now estimate the maximal number of times we need to use the division algorithm. We can assume that  $r_0 = a \geq r_1 = b$  and see that  $r_n = r_{n+1}q_{n+2} + r_{n+2} \geq r_{n+1} + r_{n+2}$ . If  $r_0 > r_1 > \cdots > r_n = d$  is the sequence which gives the gcd, set  $d_i = r_{n-i}$ . We then have  $d_{i+2} \geq d_{i+1} + d_i$ . Let  $\alpha := (1 + \sqrt{5})/2$  be the positive root of  $X^2 = X + 1$ ; it follows that  $d_i \geq \alpha^i$ . This is true because  $d_0 = d \geq 1 = \alpha^0$ ,  $d_1 \geq d_0 + 1 \geq 2 \geq \alpha^1$  and if the inequality is true until  $i+1$ , we have  $d_{i+2} \geq d_{i+1} + d_i \geq \alpha^{i+1} + \alpha^i = \alpha^{i+2}$ . From this we conclude that  $a = d_n \geq \alpha^n$ , and the number of steps is bounded above by  $\log(a)/\log(\alpha) = O(\log a)$ . We should point out that this argument implies that the longest computation happens when  $a$  and  $b$  are terms in Fibonacci sequence (see Exercise 2-7.5). The total cost is therefore  $O(\log \max\{|a|, |b|\}^3)$ .

**Computations in  $\mathbf{Z}/N\mathbf{Z}$ .** The goal is to perform addition and multiplication of two integers smaller than  $N$ , then to take the remainder gotten from dividing by  $N$  in the division algorithm. In order to calculate the inverse of  $a$  modulo  $N$ , we proceed as follows: if  $a$  is an integer, the Euclidean algorithm tells us that either  $\gcd(a, N) > 1$ —in which case  $a$  is not invertible modulo  $N$ —or there exist  $u, v$  (gotten from the algorithm) such that  $au + Nv = 1$  and therefore the inverse of  $a$  is the class of  $u$  modulo  $N$ . The cost is therefore the same as that of the Euclidean algorithm.

**Exponentiation.** In order to calculate  $a^m$ , we could of course calculate  $a \times a \times \cdots \times a$ , but this will force us to perform  $m - 1$  multiplications; we could do a lot better by performing the computation in  $O(\log m)$  multiplications. For example, if  $m = 2^r$  we would carry out  $r$  multiplications. In the general case, we write  $m$  in binary notation  $m = \epsilon_0 + \epsilon_1 2 + \cdots + \epsilon_r 2^r$  and we would calculate

$$a^m = \left( \left( (a^{\epsilon_r})^2 a^{\epsilon_{r-1}} \right)^2 a^{\epsilon_{r-2}} \cdots \right)^2 a^{\epsilon_0}.$$

Or we could do the calculation in the other direction; the algorithm can be defined iteratively. In order to do this, we start with the initial data chosen to be  $(u, v, n) := (1, a, m)$  and we iterate as follows: if  $n$  is even, we replace  $(u, v, n)$  by  $(u, v^2, n/2)$  and if  $n$  is odd, we replace  $(u, v, n)$  by  $(uv, v^2, n - 1)$ .

$1)/2$ ); we stop when  $n = 0$ , and we therefore have  $u = a^m$ . Since  $n$  is at least divisible by 2 in each step, the number of steps  $r$  satisfies  $2^r \leq m$ , and hence we must perform  $O(\log m)$  multiplications. If we calculate mod  $N$ , we reduce each result mod  $N$ , and so in each step we multiply integers  $\leq N$ . The total cost to compute  $a^m \bmod N$  is therefore  $O(\log m(\log N)^2)$ .

**Computations in  $\mathbf{F}_q$  and  $\mathbf{F}_q^*$ .** We will assume that the finite field  $\mathbf{F}_q = \mathbf{F}_{p^f}$  is defined by an irreducible monic polynomial  $S(X) = X^f + s_{f-1}X^{f-1} + \cdots + s_0 \in \mathbf{F}_p[X]$  of degree  $f$ . We therefore identify  $\mathbf{F}_q$  with  $\mathbf{F}_p[X]/S\mathbf{F}_p[X]$ , which can be seen as the vector space over  $\mathbf{F}_p$  with basis  $1, x, x^2, \dots, x^{f-1}$  with addition on the individual coordinates and multiplication defined by  $x^i \cdot x^j = x^{i+j}$  and  $x^f = -s_{f-1}x^{f-1} - \cdots - s_0$ . An element of  $\mathbf{F}_q$  is therefore seen as an  $f$ -tuple of integers modulo  $p$  or as a polynomial of degree  $\leq f-1$ . To perform an addition, we must perform  $f$  additions in  $\mathbf{F}_p$ , so at a cost of  $O(f \log p) = O(\log q)$ . To carry out a multiplication, we take the product of two polynomials, or essentially  $f^2$  multiplications in  $\mathbf{F}_p$ , then divide the result by  $S(X)$  using the division algorithm, or essentially  $O(f)$  divisions and  $O(f^2)$  multiplications in  $\mathbf{F}_p$ . The cost of a multiplication in  $\mathbf{F}_q$  is therefore  $O(f^2(\log p)^2) + O(f(\log p)^3)$ . Let us point out that this cost is still  $O((\log q)^3)$ , but that if we choose  $q = 2^f$  for example, it is  $O(f^2) = O((\log q)^2)$ .

## 2. Cryptography, RSA

*We are only interested here in one aspect of cryptography and in one system of “public keys”, known as RSA from the name of its three inventors, Rivest, Shamir and Adleman [61], and which is one of the most widely used.*

Cryptography is the art (or science) of secret messages: we want to send information so that only one other person, the recipient, can see it. A related problem is to be able to identify with certainty the sender of the message. We generally think that the only method is to use a “secret code”; in fact the originality of “public key” cryptography comes precisely from the fact that the code is not secret, but is known (for the most part) by everybody! This is not only a mathematical curiosity, it is also the principle governing credit cards, internet transactions, etc.

The general principle is the following. We call  $\mathcal{M}$  the set of messages (in practice we take  $\mathcal{M} = [0, N-1]$  or  $\mathbf{Z}/N\mathbf{Z}$ ). Two people, A and B, who wish to exchange messages in such a way that a third person, C, cannot decipher them each choose bijections  $f_A, f_B : \mathcal{M} \rightarrow \mathcal{M}$ . The set  $\mathcal{M}$  (say the integer  $N$ ) is known to everybody, as well as  $f_A$  and  $f_B$ , however—and this is the key idea—the inverse function  $f_A^{-1}$  (resp.  $f_B^{-1}$ ) is only known by A (resp. by B). This does not mean of course that, knowing  $f_A$ , it is

theoretically impossible to compute  $f_A^{-1}$ , but this calculation would be so long, that it would be out of the question to carry out in a reasonable time frame. We will later see how to construct such functions.

When A wants to send B a message  $m \in \mathcal{M}$  (say an integer modulo  $N$ ), he or she simply sends  $m' = f_B \circ f_A^{-1}(m)$ ; remember that A knows  $f_B$  (which is public) and  $f_A^{-1}$  (which only he or she knows). In order to decode this message, B computes  $f_A \circ f_B^{-1}(m')$ , which will give  $m$ ; remember that B knows  $f_A$  (which is public) and  $f_B^{-1}$  (which only he or she knows). The system has two advantages: not only can C not decipher the message without computing  $f_B^{-1}$  (which we assume to be out of the question), but B can be sure that it is A who sent the message since it must have been encoded using  $f_A^{-1}$ , which only A knows!

This procedure is a simplified form of the known methods under the name of the Diffie-Hellman protocol (1976); its security relies on the choice of the “one-way” functions  $f$ , in other words such that  $f$  is quick and easy to compute, but  $f^{-1}$  is in practice impossible to determine. Many constructions of functions have been suggested, but one of the most hardy and most widely used, relies on the fact that if  $p$  and  $q$  are very large prime numbers (say 100 or more digits), then their product  $N := pq$  can be calculated very quickly (say 10,000 elementary operations), whereas if you only know  $N$ , it is an extremely long calculation to factor it, impossible in practice.

We now construct the functions  $f_A$  of the RSA system. We choose two very large prime numbers,  $p$  and  $q$ , compute  $N := pq$  and also choose a medium-sized integer  $d$  which is relatively prime to  $\phi(N) = (p-1)(q-1)$ . The public key is therefore  $(N, d)$ ; however,  $p$  and  $q$  are secret and we set, for  $a$  any integer smaller than  $N$ ,

$$f(a) := a^d \bmod N.$$

To decode a message, we calculate the inverse  $e$  of  $d$  modulo  $\phi(N)$  and we observe that

$$f^{-1}(b) = b^e \bmod N,$$

since  $(a^d)^e = a^{ed} \equiv a \bmod N$ , because  $a^{\phi(N)} \equiv 1 \bmod N$ .

**2.1. Remarks.** 1) There is one little constraint on the “message”  $a$ : it should be relatively prime to  $N$ <sup>1</sup>. Nonetheless, observe that the proportion of integers which are relatively prime to  $N$  is  $\phi(N)/N = (1-1/p)(1-1/q)$ ; so if  $p, q$  are for example  $\geq 10^{50}$ , the proportion of integers which are not relatively prime to  $N$  is  $\leq 2 \cdot 10^{-50}$ .

---

<sup>1</sup>If by mistake, a message  $a = pa'$  was sent, we could certainly still decode it by  $f(a)^e = p^{de} a'^{ed} = p^{ed} a' = a$ , but C, or whoever else, would only have to compute  $\gcd(a, N)$  to discover  $p$  and crack the code!

2) Once  $p$ ,  $q$  and  $d$  have been chosen, the computation of  $N$ ,  $\phi(N)$  and  $e$  is performed in polynomial time (fast); likewise, the operation  $a \mapsto f(a)$  is just as fast as  $a \mapsto f^{-1}(a)$  if we know  $e$ .

3) We can see, at least heuristically, that knowing the number  $e$  allows us to factor  $N$ : if we write  $de - 1 = 2^r M$  (with  $M$  odd), by computing  $\gcd(a^{2^j M} \pm 1, N)$  for  $j = 1, 2, \dots$  and some values of  $a$ , we have a good chance of quickly factoring  $N$ .

4) Therefore, if someone knows only the public key  $(N, d)$ , they should *a priori* factor  $N$  in order to compute  $\phi(N)$  then  $e$ . In fact, the knowledge of  $\phi(N)$  is equivalent to that of  $p$  and  $q$ , because  $\phi(N) = N - (p + q) + 1$  (the knowledge of the product and the sum of two integers lets you easily determine the integer pair).

This system gives rise to many problems, the solutions to which are more or less satisfactory.

- i) How do you construct (very) large prime numbers?
- ii) What methods do we have for factoring an integer?
- iii) How should you choose  $p$  and  $q$  in RSA that resist factorization methods?

Since it is clear from question *iii*) that the prime numbers should not be too “special”, question *i*) is essentially equivalent to the following problem.

- (I) (Primality Test) Give a fast algorithm which determines whether a number  $N$  is prime.

If we had access to such an algorithm  $\mathcal{P}$ , we could in fact decide on the size of the integer (for example  $N \sim 10^{50}$ ), randomly choose an odd integer  $N_1$  of this size, and test  $\mathcal{P}(N_1)$  then  $\mathcal{P}(N_1 + 2)$ ,  $\mathcal{P}(N_1 + 4)$  until we find a prime number. By the theorems on the distribution of prime numbers, the number of primes in an interval  $[N_1, N_1 + H]$  is approximately  $H / \log(N_1)$ ; so we expect to find a prime number in  $O(\log(N_1))$  tries.

We will see that satisfactory answers to problem *i*) are available, but we only know partial answers to the other questions.

### 3. Primality Test (I)

We consider an *odd* integer  $N$  and the problem of determining whether  $N$  is prime. We denote by  $(M, N)$  the gcd of  $M$  and  $N$ . The letter  $p$  is reserved for a number which we already know is prime. The first of all of the primality tests, and in some sense the “grandfather”, is the following lemma.

**3.1. Lemma.** (Fermat) *If  $N$  is prime and  $(a, N) = 1$ , then  $a^{N-1} \equiv 1 \pmod{N}$ .*

*Proof.* The group  $\mathbf{Z}/N\mathbf{Z}^*$  has order  $N - 1$  and the lemma follows from the Lagrange's theorem.<sup>2</sup>  $\square$

This is a “good” test, in the sense that computing  $a^{N-1} \pmod{N}$  requires  $O(\log N)$  multiplications (under the condition of course that you use the binary notation for  $N - 1$ ). However, it is also a “bad” test, because there are numbers, called *Carmichael numbers*, which satisfy the test without being prime. We even know that there are infinitely many of them [11], the smallest being  $561 = 3 \cdot 11 \cdot 17$ . We can easily see that a number  $N$  is a Carmichael number if and only if  $N$  is square-free and  $p - 1$  divides  $N - 1$  for every  $p$  which divides  $N$ . In general, we could introduce  $\lambda(N)$ , the exponent of the group  $(\mathbf{Z}/N\mathbf{Z})^*$ , sometimes called the *Carmichael function*: it is the smallest positive integer (in the sense of divisibility or the usual order) such that for all  $a$  relatively prime to  $N$ ,  $a^{\lambda(N)} \equiv 1 \pmod{N}$ . By what we have seen, we know that if  $N = p_1^{m_1} \cdots p_k^{m_k}$  is odd, we have

$$\lambda(N) = \text{lcm}(p_1^{m_1-1}(p_1 - 1), \dots, p_k^{m_k-1}(p_k - 1)). \quad (2.1)$$

It is always true that  $\lambda(N)$  divides  $\phi(N)$  and the equality holds if and only if  $(\mathbf{Z}/N\mathbf{Z})^*$  is cyclic, i.e., if  $N = p^\alpha$  or  $2p^\alpha$  or 4.

**3.2. Lemma.** (Euler<sup>3</sup>) *If  $N$  is prime and  $(a, N) = 1$ , then*

$$a^{\frac{N-1}{2}} \equiv \left( \frac{a}{N} \right) \pmod{N}.$$

*Proof.* This is simply a restatement of assertion *ii*) from Theorem 1-3.3.  $\square$

The *Solovay-Strassen test* is an algorithm which checks the congruences given below for a randomly chosen  $a$ . This test is always polynomial (for any value of  $a$ , we can always quickly calculate the Jacobi symbol thanks to the quadratic reciprocity law, see Exercise 2-7.7) and is better than Fermat's test.

**3.3. Lemma.** *Let  $H := \left\{ a \in (\mathbf{Z}/n\mathbf{Z})^* \mid a^{\frac{N-1}{2}} \equiv \left( \frac{a}{N} \right) \pmod{N} \right\}$ , then  $H = (\mathbf{Z}/n\mathbf{Z})^*$  if and only if  $N$  is a prime number.*

---

<sup>2</sup>To prove Fermat's little theorem by using Lagrange's theorem is obviously an anachronism.

<sup>3</sup>Calling a statement which uses the Legendre or Jacobi symbol “Euler's criterion” is also an anachronism.

*Proof.* We have seen that if  $N$  is prime, then  $H = (\mathbf{Z}/n\mathbf{Z})^*$ . If  $p^2$  divides  $N$ , there exists  $a$  of order  $p(p-1)$ , and  $p$  does not divide  $N-1$ . Therefore,  $a^{N-1} \neq 1$ . If  $N = pp_2 \cdots p_r$  with  $r \geq 2$ , choose (by the Chinese remainder theorem)  $a \equiv 1$  modulo  $p_2, \dots, p_r$  and which is not a square modulo  $p$ ; hence  $\left(\frac{a}{N}\right) = -1$ , but  $a^{(N-1)/2} \equiv 1 \pmod{p_2 \cdots p_r}$  and thus  $a^{(N-1)/2} \not\equiv -1 \pmod{N}$ .  $\square$

### Applications.

- i) Probabilistic polynomial test. If  $N$  is composite then  $(\mathbf{Z}/N\mathbf{Z}^* : H) \geq 2$  and hence by randomly choosing  $a$ , we have at least a one in two chance that  $a \notin H$ . Hence if  $N$  successively passes  $k$  tests, we can say that it is prime with a probability greater than  $1 - 2^{-k}$ .
- ii) Deterministic polynomial test (assuming GRH). Analytic theory has provided a proof that if the Dirichlet  $L(\chi, s)$  functions do not vanish on  $\text{Re}(s) > 1/2$  (generalized Riemann hypothesis, GRH), then for every nontrivial character  $\chi : (\mathbf{Z}/N\mathbf{Z})^* \rightarrow \mathbf{C}^*$ , there exists an  $a \leq 2(\log N)^2$  such that  $\chi(a) \neq 0, 1$ . We can deduce from this that if  $N$  were composite, there would exist  $a \leq 2(\log N)^2$  which would not pass the Solovay-Strassen test. If  $N = p_1^{m_1} \cdots p_k^{m_k}$ , we introduce  $f(a) := a^{\frac{N-1}{2}} \left(\frac{a}{N}\right)$  and

$$\chi_i : (\mathbf{Z}/N\mathbf{Z})^* \xrightarrow{f} (\mathbf{Z}/N\mathbf{Z})^* \rightarrow (\mathbf{Z}/p_i^{m_i}\mathbf{Z})^* \hookrightarrow \mathbf{C}^*.$$

We see that  $H$  is the intersection of the kernels of  $\chi_i$ . By trying all of the  $a \in [2, 2(\log N)^2]$ , we therefore get a primality certificate (i.e., a proof of primality), under the condition that the Riemann hypothesis is true.

We could improve the Solovay-Strassen test and algorithm.

**3.4. Lemma.** (*Rabin-Miller*) Let  $N$  be odd. Set  $N-1 = 2^s M$ , with  $M$  odd. If  $N$  is prime and  $(a, N) = 1$ , then either  $a^M \equiv 1 \pmod{N}$  or there exists  $0 \leq r \leq s-1$  such that  $a^{2^r M} \equiv -1 \pmod{N}$ .

*Proof.* The order of  $a$  modulo  $N$  is  $2^t M'$ , where  $0 \leq t \leq s$  and  $M'$  is an odd integer which divides  $M$ . If  $t = 0$ , then  $a^{M'} = 1$  hence  $a^M = 1$ . If  $t \geq 1$ , then, since  $N$  is prime,  $a^{2^{t-1} M'} = -1$ , and therefore  $a^{2^{t-1} M} = -1$ .  $\square$

This test is better than Euler's test, because, for one thing, if the pair  $a, N$  passes the Rabin-Miller test, then it also must pass Euler's test. Furthermore, if  $N$  is composite, the proportion of  $a$  which pass the refined test



is  $\leq 1/4$  and often smaller than that. Of course there exists a probabilistic polynomial version of the refined test and a deterministic polynomial version, assuming that the Riemann hypothesis is true.

**3.5. Remark.** If  $N \equiv 3 \pmod{4}$ , then “Rabin-Miller” is identical to “Solovay-Strassen”, and even equivalent to  $a^{(N-1)/2} \equiv \pm 1 \pmod{N}$ . We know that  $(N-1)/2$  is odd, and we can observe that if  $\epsilon = \pm 1$ , then  $\left(\frac{\epsilon}{N}\right) = \epsilon$ , and if  $a^{(N-1)/2} \equiv \pm 1 \pmod{N}$ , then

$$\left(\frac{a}{N}\right) = \left(\frac{a \cdot (a^2)^{(N-3)/4}}{N}\right) = \left(\frac{a^{(N-1)/2}}{N}\right) = a^{(N-1)/2} \pmod{N}.$$

*Proof.* (“Rabin-Miller”  $>$  “Solovay-Strassen”, in the general case) Now, we know that  $a^{(N-1)/2} = a^{2^{s-1}M}$  equals  $-1 \pmod{N}$  if  $r = s-1$  and equals  $1 \pmod{N}$  in all of the other cases. Therefore, we need to compute  $\left(\frac{a}{N}\right)$ . If  $a^M \equiv 1 \pmod{N}$ , then  $\left(\frac{a}{N}\right) = \left(\frac{a}{N}\right)^M = \left(\frac{a^M}{N}\right) = 1$ , hence  $a^{\frac{N-1}{2}} \equiv \left(\frac{a}{N}\right) \pmod{N}$ . Now assume that  $a^{2^r M} \equiv -1 \pmod{N}$ . Let  $p_i$  divide  $N$  and write  $p_i - 1 = 2^{s_i} M_i$ . Then, since  $a^{2^r M} \equiv -1 \pmod{p_i}$ , the order of  $a$  modulo  $p_i$  is of the form  $2^{r+1} L_i$  (with  $L_i$  odd). Therefore, modulo  $p_i$ , we get

$$\left(\frac{a}{p_i}\right) \equiv a^{(p_i-1)/2} \equiv a^{2^{s_i-1} M_i} \equiv \begin{cases} 1 & \text{if } s_i > r+1, \\ -1 & \text{if } s_i = r+1. \end{cases}$$

Now notice that  $r+1 \leq s_i$ . Let  $h$  be the number of indices  $i$  such that  $s_i = r+1$ . Therefore, we have  $\left(\frac{a}{N}\right) = (-1)^h$ . Modulo  $2^{r+2}$ , we have  $N = 1 + 2^s M = \prod_i p_i = \prod_i (1 + 2^{s_i}) \equiv 1 + h2^{r+1} \pmod{2^{r+2}}$ . In the case where  $r < s-1$ ,  $h$  must be even, so that  $\left(\frac{a}{N}\right) = 1$ , and we get  $a^{(N-1)/2} \equiv 1 \pmod{N}$ . In the case where  $r = s-1$ , then  $h$  is odd and  $\left(\frac{a}{N}\right) = -1 \equiv a^{(N-1)/2} \pmod{N}$ .  $\square$

We can summarize the previous discussion by introducing the following sets:

$$G_0 := (\mathbf{Z}/N\mathbf{Z})^*,$$

$$G_1 := \{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^{N-1} \equiv 1 \pmod{N}\},$$

$$G_2 := \{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^{(N-1)/2} \equiv \pm 1 \pmod{N}\},$$

$$G_3 := \left\{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^{(N-1)/2} \equiv \left(\frac{a}{N}\right) \pmod{N}\right\},$$

$$S := \{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^M \equiv 1 \pmod{N} \text{ or } \exists r \in [0, s-1] \text{ such that } a^{2^r M} \equiv -1 \pmod{N}\}.$$

We always have the inclusions  $S \subset G_3 \subset G_2 \subset G_1 \subset G_0$ , and these are equalities if and only if  $N$  is prime, or also if and only if  $G_3 = G_0$ . Furthermore,  $G_1$ ,  $G_2$  and  $G_3$  are subgroups, but in general  $S$  is not, even though in the case  $N \equiv 3 \pmod{4}$  we have seen that  $G_2 = G_3 = S$ . In fact,  $S$  is stable under inversion, and if  $a, b \in S$  do not satisfy the same congruence or both  $a^M = b^M = 1$ , then  $ab \in S$ . But if  $a^{2^r M} = b^{2^r M} = -1$ , it could happen that  $ab \notin S$ . For example, if  $\epsilon^2 = 1$  but  $\epsilon \neq \pm 1$  and if  $a^{2^M} = -1$  (which would force  $N \equiv 1 \pmod{4}$ ), then  $a \in S$  and  $a\epsilon \in S$ , because  $(a\epsilon)^{2^M} = -1$ . However,  $(\epsilon a^2)^M = \epsilon^M a^{2^M} = -\epsilon \neq \pm 1$  and  $(\epsilon a^2)^{2^M} = 1$ , hence  $\epsilon a^2 \notin S$ . By considering  $a \mapsto \left(\frac{a}{N}\right) a^{(N-1)/2}$  from  $G_2$  to  $\{\pm 1\}$ , we see that  $(G_2 : G_3) = 1$  or  $2$ . We are now going to compute the cardinality of the set  $S$  and, in particular, verify the following statement.

**3.6. Proposition.** *Let  $N$  be an odd, composite number. If  $N \neq 9$ , then*

$$\frac{|S|}{|G_0|} \leq \frac{1}{4}.$$

**3.7. Definition.** Let  $A, B$  be integers. We define

$$\phi(A; B) = \text{card} \{a \in (\mathbf{Z}/A\mathbf{Z})^* \mid a^B \equiv 1 \pmod{A}\}.$$

**3.8. Lemma.** *Let  $t \geq 0$  and  $N = 1 + 2^s M = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$  (with  $M$  odd). We set  $p_i - 1 = 2^{s_i} M_i$ ,  $s'_i = \min(t, s_i)$  and  $t_i := \gcd(M, M_i)$ . Then*

$$\phi(N, 2^t M) = 2^{s'_1 + \cdots + s'_k} t_1 \cdots t_k.$$

Moreover, the cardinality of the set

$$\left\{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^{2^t M} \equiv -1 \pmod{N}\right\}$$

is 0 if  $t \geq \min_i s_i$ , and equal to  $\phi(N, 2^t M) = 2^{t_k} t_1 \cdots t_k$  if  $t < \min_i s_i$ .

*Proof.* We know that  $a^{2^t M} \equiv 1 \pmod{N}$  if and only if  $a^{2^t M} \equiv 1 \pmod{p_j^{\alpha_j}}$  for  $j = 1, \dots, k$ . Now, the group  $(\mathbf{Z}/p_j^{\alpha_j} \mathbf{Z})^*$  is cyclic of order  $(p_j - 1)p_j^{\alpha_j - 1}$ , so the number of solutions is

$$\gcd(2^t M, (p_j - 1)p_j^{\alpha_j - 1}) = \gcd(2^t M, 2^{s_j} M_j) = 2^{\min(t, s_j)} t_j.$$

By the Chinese remainder theorem, the number of solutions modulo  $N$  is therefore the product of these numbers, and hence we have proven the first claim. For the second claim, we see right away that either there does

not exist any solution, or there does exist a solution and therefore the set of solutions is in bijection with the solutions of the previous congruence. The congruence  $a^{2^t M} \equiv -1 \pmod{p_j^{\alpha_j}}$  is solvable if and only if  $2^{t+1}$  divides  $(p_j - 1)p_j^{\alpha_j - 1}$ , in other words if and only if  $t + 1 \leq s_j$ , hence we have the desired result.  $\square$

*Proof.* (of Proposition 2-3.6) Assume that  $s_1 \leq s_2 \leq \dots \leq s_k$ . By decomposing the set  $S$  into  $S_0 := \{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^M \equiv 1 \pmod{N}\}$  and  $T_j := \{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid a^{2^j M} \equiv -1 \pmod{N}\}$  for  $0 \leq j \leq s_1 - 1$  and by applying Lemma 2-3.8 to each one of these sets, we have

$$\text{card}(S) = t_1 \cdots t_k \left(1 + 1 + 2^k + \dots + 2^{k(s_1-1)}\right) = t_1 \cdots t_k \left(\frac{2^{ks_1} + 2^k - 2}{2^k - 1}\right).$$

The ratio of  $a \in G_0$  which pass the Rabin-Miller test is therefore

$$\frac{\text{card}(S)}{\text{card}(G_0)} = \frac{t_1 \cdots t_k}{M_1 \cdots M_k} \frac{2^{-(s_1 + \dots + s_k)}}{p_1^{\alpha_1 - 1} \cdots p_k^{\alpha_k - 1}} \left(\frac{2^{ks_1} + 2^k - 2}{2^k - 1}\right). \quad (2.2)$$

If  $k = 1$ , the ratio is equal to  $\frac{t_1}{M_1 p_1^{\alpha_1 - 1}} \leq \frac{1}{p_1^{\alpha_1 - 1}}$ , and is therefore  $\leq \frac{1}{5}$ , except when  $N = 3^2$  in which case we have  $|S|/|G_0| = 1/3$ . If  $k \geq 2$ , we can assume that  $\alpha_1 = \dots = \alpha_k = 1$ , if not, the ratio is  $\leq 1/p_i$ , which in practice we can assume to be arbitrarily small. If one of the  $M_i$  is different from  $t_i$ , then  $t_1 \dots t_k / M_1 \dots M_k \leq 1/3$ . Furthermore,

$$2^{-s_1 - \dots - s_k} \left(\frac{2^{ks_1} + 2^k - 2}{2^k - 1}\right) \leq 2^{-ks_1} \frac{2^k - 2}{2^k - 1} + \frac{1}{2^k - 1} \leq 2^{1-k},$$

so the ratio is  $\leq 1/8$  if  $k \geq 4$  and  $\leq 1/4$  if  $k = 3$ .

If  $k = 2$  and if one of the  $M_i$  is distinct from all of the  $t_i$ , then the ratio is  $\leq 1/6$ . If  $k = 2$  and  $M_1 = t_1$  (i.e.,  $M_1$  divides  $M$ ) and  $M_2 = t_2$  (i.e.,  $M_2$  divides  $M$ ), we see that  $M_1 = M_2$ , hence  $s_1 < s_2$  (if not  $p_1 = p_2$ ). We then have that the ratio is  $\leq 2^{s_1 - s_2} \frac{1 + 2^{1-2s_1}}{3} \leq \frac{1 + 2^{1-2s_1}}{6} \leq \frac{1}{4}$ .  $\square$

**3.9. Remark.** By looking at the upper bounds above, we can prove that the two “worst” cases are the following.

- i) The number  $N$  is equal to  $pq$  with  $q = 2p - 1$  and  $p \equiv 3 \pmod{4}$ . For example,  $N = 3 \cdot 5$ ,  $N = 7 \cdot 13$ , etc. It follows that  $p = 1 + 2M_1$  and  $q = 1 + 4M_1$  and  $N = (1 + 2M_1)(1 + 4M_1) = 1 + 2M_1(3 + 4M_1)$ , hence  $t_1 = t_2 = M_1 = M_2$  and so

$$\frac{\text{card}(S)}{\text{card}(G_0)} = \frac{1}{4}.$$

- ii) The number  $N$  is equal to  $pqr = 1 + 2M$ , where  $p = 1 + 2M_1$ ,  $q = 1 + 2M_2$ ,  $r = 1 + 2M_3$  and  $M_i$  divides  $M$ . It follows from this that the ratio is also  $1/4$ . Take for example:  $N = 8911 = 7 \cdot 19 \cdot 67$  (where  $M_1 = 3$ ,  $M_2 = 9$ ,  $M_3 = 33$  and  $M = 4455 = 3^4 \cdot 5 \cdot 11$ ).

## 4. Primality Test (II)

*In this section we present the Agrawal-Kayal-Saxena algorithm [10], which dates back to July 2002, and was introduced in their article “PRIMES is in P”. It gives a primality test in polynomial time.*

The original idea was to perform tests in  $\mathbf{Z}[X]$ . For example, we easily see that if  $N$  is prime, then  $(X - a)^N \equiv X^N - a \pmod{N}$ , but this test has the major default of requiring the computation of  $N$  coefficients. That will just not do!

**4.1. Lemma.** *Let  $N$  be prime and  $h(X) \in \mathbf{Z}[X]$  a polynomial of degree  $r$ . Then*

$$(X - a)^N \equiv X^N - a \pmod{(N, h(X))}.$$

Recall that in a ring, the notation  $a \equiv b \pmod{I}$  means that  $a - b$  belongs to the ideal  $I$  and that  $(a_1, \dots, a_m)$  is the notation used for the ideal generated by  $a_1, \dots, a_m$ . Thus the congruence in the lemma can be restated as: there exists  $P, Q \in \mathbf{Z}[X]$  such that  $(X - a)^N - (X^N - a) = NP(X) + h(X)Q(X)$ .

It should be noted that if  $r$  is  $O((\log N)^k)$ , then this test remains polynomial. The problem is to choose pairs  $a, h(X)$  in such a way that they detect non-primality. The solution proposed by Agrawal, Kayal and Saxena is to choose  $h(X) = X^r - 1$  with  $r$  being a “very well-chosen” prime, in particular  $r = O((\log N)^k)$ , and to prove that it is then sufficient to test the  $a \in [1, L]$  with  $L = O(\sqrt{r} \log N)$  in order to ensure that  $N$  is prime, or possibly a prime power, which is not so bad.

The argument is essentially algebraic and combinatorial, but nevertheless uses a result on the distribution of prime numbers, in fact a weak form of the prime number theorem (see Chap. IV, (4.10)), which says that the sum of the  $\log p$  for  $p$  prime and smaller than  $x$  is  $\geq c_1 x$  for some constant  $c_1 > 0$ . We summarize what we are going to use in a lemma.

**4.2. Lemma.** *Let  $Y > 1$  and let  $N \geq 2$  be an integer. There exists a prime number  $r$  which satisfies the following two properties.*

- i) *The order of  $N$  modulo  $r$  is at least  $Y$ .*
- ii) *Furthermore,  $r = O(Y^2 \log N)$ .*

*Proof.* Set  $A := \prod_{1 \leq y \leq Y} (N^y - 1)$ . Let  $r$  be the smallest prime number which does not divide  $A$ . Then for  $y \leq Y$ , we have  $N^y \not\equiv 1 \pmod{r}$ , and hence condition *i*). Moreover, every  $p < r$  divides  $A$ , whereas  $A \leq N^{Y(Y+1)/2}$  and consequently

$$c_1 r \leq \sum_{p < r} \log p \leq \log A \leq \frac{Y(Y+1)}{2} \log N.$$

From this we have that  $r = O(Y^2 \log N)$ . □

Remark. We could add that, since the order of  $N$  modulo  $r$  divides  $r - 1$ , we necessarily have  $r > Y$ .

We will also use the following elementary combinatorial lemma.

**4.3. Lemma.** *The cardinality of the set of monomials in  $L$  variables of degree  $\leq k$  is*

$$\text{card} \{(m_1, \dots, m_L) \mid m_i \geq 0 \text{ and } m_1 + \dots + m_L \leq k\} = \binom{L+k}{k}.$$

Furthermore, we have the estimate

$$\binom{L+k}{k} \geq 2^{\min(L,k)}.$$

*Proof.* The first formula is classical and can be proven, for example, by induction (call the cardinality in question  $f(L, k)$ , check that  $f(L, 0) = 1$  and  $f(1, k) = k + 1$ , and then prove that  $f(L, k) = f(L, k-1) + f(L-1, k)$ ). For the lower bound, observe that if  $k \leq L$ , then

$$\binom{L+k}{k} = \frac{(L+k)(L+k-1) \cdots (L+2)(L+1)}{k(k-1) \cdots 2 \cdot 1} = \prod_{i=0}^{k-1} \left( \frac{L+k-i}{k-i} \right) \geq 2^k,$$

and if  $L \geq k$ , reverse the roles of  $L$  and  $k$ . □

Remark. We can often improve this inequality; for example, if  $1 \leq k \leq L$ , then  $\binom{L+k}{k} \geq 2^k(L+1)/2$ , and thus if  $L \geq 5$ , we have  $\binom{L+k}{k} \geq 2^{k+1}$ .

We will now state a version of the main theorem of Agrawal-Kayal-Saxena.

**4.4. Theorem.** *Let  $N \geq 2$  and let  $r$  be a prime number satisfying:*

- i) no prime number  $\leq r$  divides  $N$ ;*
- ii) we have  $\text{ord}(N \bmod r) \geq (2 \log N / \log 2)^2 + 1$ ;*

iii) for  $1 \leq a \leq r-1$ , we have

$$(X-a)^N \equiv X^N - a \pmod{(N, X^r-1)}.$$

Then  $N$  is a prime power.

Remarks. In order to prove this theorem, we only assume that hypothesis iii) is satisfied for  $1 \leq a \leq L$  and we will see that we can take  $L$  smaller than  $r-1$ . By Lemma 2-4.2, we can choose  $r = O((\log N)^5)$  such that ii) is satisfied, and it would necessarily follow that  $r \geq (2 \log N / \log 2)^2 + 1$ . Thus it is clear that the theorem implies that the following algorithm is correct and polynomial.

**ALGORITHM.** [10] We put in  $N$  and the algorithm returns “Prime” or “Composite”.

- 1) We check to see if  $N = a^b$  where  $b \geq 2$ ; if so, then  $N$  is “Composite”.
- 2) We try the prime numbers  $r = 2, 3, \dots$ . If  $r$  divides  $N$ ,  $N$  is “Composite”. If not, we check whether  $r$  is relatively prime  $N^y - 1$  for  $y = 1, 2, \dots, Y$ , where  $Y = \lfloor (2 \log N / \log 2)^2 \rfloor + 1$ ; if so we keep  $r$  and go to the next step, if not we look for a larger  $r$ .
- 3) For  $a = 1, 2, 3, 4, \dots$  (stop at  $r-1$ ), we check whether  $(X-a)^N \not\equiv X^N - a \pmod{(N, X^r-1)}$ . If so, then  $N$  is “Composite”, if not, we proceed to  $a+1$ .
- 4) If the algorithm keeps going until  $a = r-1$ , then  $N$  is “Prime”.

Let us briefly discuss its complexity (without trying to optimize it). We easily see that the longest step is step (3), which requires  $O(r \log N)$  multiplications in the ring  $\mathbf{Z}[X]/(N, X^r-1)$ , where each one uses at most  $O((r \log N)^2)$  elementary operations. We thus have  $O((r \log N)^3)$  in all. If we add that  $r = O((\log N)^5)$ , we obtain a complexity of at most  $O((\log N)^{18})$ .

We now proceed to the proof of the theorem. Let  $p$  be a prime divisor of  $N$ . We denote by  $d_1 := \text{ord}(N \bmod r)$ ,  $d_2 = \text{ord}(p \bmod r)$  and  $d := \text{lcm}(d_1, d_2)$ . It should be noted that  $d_1$  (resp.  $d_2$ ) is the order of the subgroup generated by  $N$  (resp. by  $p$ ) in  $(\mathbf{Z}/r\mathbf{Z})^*$  and that  $d$  is therefore the order of the subgroup generated by  $N$  and  $p$  in  $(\mathbf{Z}/r\mathbf{Z})^*$ . We then choose  $h(X)$  to be an irreducible factor of  $\Phi_r(X) := (X^r-1)/(X-1)$  in  $\mathbf{F}_p[X]$ . Let us point out, even if we do not need it, that  $\deg(h) = d_2$  (see Theorem 2-6.2.8). We will work in the field  $K := \mathbf{F}_p[X]/(h(X))$ , which is a finite field (isomorphic to  $\mathbf{F}_{p^{d_2}}$ ) and which we obtain by adding a primitive  $r$ th root of unity to  $\mathbf{F}_p$ . By construction,  $x := X \bmod h(X)$  is of order  $r$  in  $K^*$ . It is natural to look at the subgroup  $G$  of  $K^*$  generated by the classes of  $(X-a)$  for  $1 \leq a \leq L$ . The heart of the proof consists of finding an upper and lower bound for the order of  $G$ .

**4.5. Lemma.** *We have the lower bound*

$$\text{card}(G) \geq \binom{L+d-1}{d-1} \geq 2^{\min(L, d-1)}.$$

From the remark immediately following the combinatorial lemma (Lemma 2-4.3), we have for example that if  $1 \leq d-1 \leq L$ , then  $\text{card}(G) \geq 2^d$ , and if  $L \leq d$ , then  $\text{card}(G) \geq 2^{L+1}$ .

*Proof.* In light of the combinatorial lemma mentioned above, it suffices to show that the classes of elements,

$$\prod_{1 \leq a \leq L} (X - a)^{m_a}, \quad \text{for } m_a \geq 0 \quad \text{and} \quad \sum_{a=1}^L m_a \leq d-1,$$

are all distinct in  $K$ . First of all, the  $a$  are distinct modulo  $p$ , because if not, then  $p \leq L < r$  and we assumed that  $N$  was not divisible by any prime number smaller than  $r$ , so  $p > r$ . Thus our polynomials are all distinct in  $\mathbf{F}_p[X]$ . Now we bring in the key point that if  $P = \prod_{1 \leq a \leq L} (X - a)^{m_a}$ , then we have, on one hand,  $P(X)^N \equiv P(X^N) \pmod{(N, X^r - 1)}$ , but also  $P(X)^p \equiv P(X^p) \pmod{p}$ , so the two congruences are valid  $\pmod{(p, X^r - 1)}$ . For  $m = N^i p^j$ , it therefore follows that

$$P(X)^m \equiv P(X^m) \pmod{(p, X^r - 1)} \quad \text{or even} \quad \pmod{(p, h(X))}.$$

In fact, the set of  $m$  such that  $P(X)^m \equiv P(X^m) \pmod{(p, X^r - 1)}$  is multiplicative (the fairly simple proof is given in detail in part *ii*) of 2-4.7 below). Now let  $P$  and  $Q$  be two polynomials of the form given above (considered in  $\mathbf{F}_p[X]$ ), and suppose that they are in the same class in  $K$ , i.e., suppose  $P \equiv Q \pmod{(p, h(X))}$ . Let  $x$  be the class of  $X$ , which is an  $r$ th primitive root of unity in  $K$ , and therefore

$$(P - Q)(x^m) = 0, \quad \text{for } m \in \langle N, p \rangle \subset (\mathbf{Z}/r\mathbf{Z})^*.$$

But we know that  $N$  and  $p$  generate a subgroup of order  $d$  in  $(\mathbf{Z}/r\mathbf{Z})^*$ , thus the polynomial  $P - Q$  has at least  $d$  roots, and since  $\deg(P - Q) \leq d - 1$ , we see that  $P = Q$  (first in  $\mathbf{F}_p[X]$ , then, if we want, in  $\mathbf{Z}[X]$ ).  $\square$

In order to find an upper bound for  $|G|$ , we choose a generator of  $G$  (it is a subgroup of  $K^*$  and is thus cyclic) and define the following set.

**4.6. Definition.** Let  $g$  be a generator of  $G$ . We define

$$\mathcal{J} = \mathcal{J}_g := \{m \in \mathbf{N} \mid g(X)^m \equiv g(X^m) \pmod{(X^r - 1, p)}\}.$$

The main properties of  $\mathcal{J}$  are summarized in the following lemma.

**4.7. Lemma.** *The set  $\mathcal{S}$  satisfies the following properties.*

- i)  $N$  and  $p$  are in  $\mathcal{S}$ .
- ii)  $\mathcal{S}$  is multiplicative, i.e., if  $m_1$  and  $m_2 \in \mathcal{S}$ , then  $m_1 m_2 \in \mathcal{S}$ .
- iii) If  $m_1$  and  $m_2 \in \mathcal{S}$  satisfy  $m_1 \equiv m_2 \pmod{r}$ , then  $m_1 \equiv m_2 \pmod{\text{card}(G)}$ .

*Proof.* The first property has already been established. For ii), write

$$g(X)^{m_1 m_2} = (g(X)^{m_1})^{m_2} \equiv (g(X^{m_1}))^{m_2} \pmod{p, X^r - 1},$$

and notice that since  $m_2 \in \mathcal{S}$ , we have  $g(Y)^{m_2} \equiv g(Y^{m_2}) \pmod{p, Y^r - 1}$ . Therefore, by substituting  $Y = X^{m_1}$ , we obtain

$$\begin{aligned} (g(X^{m_1}))^{m_2} &= g(X^{m_1 m_2}) + pQ_1(X^{m_1}) + (X^{m_1 r} - 1)Q_2(X^{m_1}) \\ &\equiv g(X^{m_1 m_2}) \pmod{p, X^r - 1}. \end{aligned}$$

In order to prove iii), suppose that  $m_1$  and  $m_2 \in \mathcal{S}_g$  and that  $m_2 = m_1 + kr$ , where  $k \geq 0$ . It follows from this that

$$g(X)^{m_2} \equiv g(X^{m_2}) \pmod{X^r - 1, p} \quad \text{and thus} \quad \pmod{h(X), p};$$

hence  $g(X)^{m_1 + kr} = g(X^{m_1 + kr})$  in  $K$ . But  $X^{m_1 + kr} \equiv X^{m_1} \pmod{X^r - 1}$  and therefore  $\pmod{h(X)}$ . Thus we obtain the equality in  $K^*$

$$g(X)^{m_1} g(X)^{kr} = g(X^{m_1}) = g(X)^{m_1},$$

where the last equality comes from the hypothesis that  $m_1 \in \mathcal{S}$ . From this, we of course have that  $g(X)^{kr} = 1 \in K^*$  and hence  $\text{card}(G)$  divides  $kr = m_2 - m_1$ .  $\square$

*Proof.* (end of the proof of Theorem 2-4.4) In order to apply the lemma, we use that  $N$ ,  $p$  and hence all of the products of powers  $N^i p^j$  are in  $\mathcal{S}$ . Recall that these elements generate a subgroup of order  $d$  in  $(\mathbf{Z}/r\mathbf{Z})^*$ . If we set

$$E := \{(i, j) \in \mathbf{N} \times \mathbf{N} \mid 0 \leq i, j \leq \sqrt{d}\},$$

then the cardinality of  $E$  is  $(\lfloor \sqrt{d} \rfloor + 1)^2 > d$ . By the pigeonhole principle<sup>4</sup>, there are two elements  $N^{i_1} p^{j_1}$  and  $N^{i_2} p^{j_2}$ , which are congruent modulo  $r$ , and such that  $(i_1, j_1)$  and  $(i_2, j_2)$  are distinct in  $E$ . These two elements  $N^{i_1} p^{j_1}$  and  $N^{i_2} p^{j_2}$  are therefore congruent modulo  $\text{card}(G)$ . First suppose that  $N^{i_1} p^{j_1} \neq N^{i_2} p^{j_2}$ , which implies that

$$\text{card}(G) \leq |N^{i_1} p^{j_1} - N^{i_2} p^{j_2}| \leq N^{2\sqrt{d}}.$$

If we combine this upper bound with the lower bound gotten above, we see that

$$\min(L + 1, d) \log 2 \leq (2\sqrt{d}) \log N.$$

---

<sup>4</sup>The pigeonhole principle says that if we put  $n + 1$  pigeons into  $n$  boxes, at least one of the boxes will contain at least two pigeons.



We will prove that this inequality is impossible.

- 1) If we had  $L \geq d$ , we could deduce that  $\sqrt{d} \leq 2 \log N / \log 2$  or moreover that  $d \leq (2 \log N / \log 2)^2$ . But this inequality is a contradiction since, by construction,  $d \geq d_1$ , and we assumed that  $d_1 > (2 \log N / \log 2)^2$ .
- 2) Now if  $L < d$ , we deduce that  $(L + 1) \log 2 \leq (2\sqrt{d}) \log N$ , and since  $d \leq r - 1$ , this would give us  $(L + 1) \log 2 \leq 2\sqrt{r - 1} \log N$ .

It is therefore a sufficient condition that  $L \geq 2\sqrt{r - 1} \log N / \log 2$  is large enough in order to conclude that  $N^{i_1} p^{j_1} = N^{i_2} p^{j_2}$ . The choice  $L = r - 1$  is suitable<sup>5</sup> since then the desired equality would be equivalent to the inequality  $\sqrt{r - 1} \geq 2 \log N / \log 2$ , which is where the hypothesis  $r \geq (2 \log N / \log 2)^2 + 1$  comes from. We finish the proof by pointing out that the inequality  $N^{i_1} p^{j_1} = N^{i_2} p^{j_2}$  immediately implies that  $N = p^\alpha$ .  $\square$

**4.8. Remark.** One variation of this proof consists of abandoning the constraint that  $r$  is a prime number; we choose a factor,  $h(X)$ , of  $\Phi_r \in \mathbf{F}_p[X]$  where  $\Phi_r$  is the  $r$ th cyclotomic polynomial (cf. Sect. 6 of this chapter), and we could then omit every analytic estimate of the distribution of prime numbers (see [33] for this version, as well as a finer estimate of the complexity).

## 5. Factorization

We briefly consider, and necessarily very unsatisfactorily, the problem of factorization: having established, by a primality test, that an integer  $N$  is not prime, how could we go about factoring it? We start by pointing out that the (complete) factorization problem is essentially equivalent to the problem of finding one factor, because of course, by iterating this procedure, we would achieve a complete factorization.

The naive factorization method consists of checking if 2 divides  $N$ , then if 3 divides  $N$ , etc. If  $N = pq$  where  $p$  and  $q$  are roughly of the same size, i.e.,  $p \sim q \sim \sqrt{N}$ , we see that we would need to perform  $O(\sqrt{N})$  divisions before arriving at a factorization of  $N$ . The naive algorithm is thus exponential.

There do exist more efficient algorithms. In fact, one of the best algorithms known [49] (using elliptic curves) has a number of operations estimated by  $\exp(C\sqrt{\log p \log \log p})$ , where  $p$  is the smallest prime factor of  $N$ . In the case where  $N = pq$  where  $p \sim q \sim \sqrt{N}$ , we therefore get an algorithm with an order of complexity  $\exp(C'(\log N)^\kappa)$  (where  $\kappa < 1$ ), which grows

---

<sup>5</sup>We point out however that we could take  $L = O(\sqrt{r} \log N)$ , which would allow us to slightly improve the estimate of the complexity.

less quickly than  $N^\kappa$  but more quickly than  $(\log N)^\kappa$ . We say that such an algorithm is *subexponential*. Another algorithm [19] (“number field sieve”) has a complexity on the order of  $\exp(C(\log N)^{1/3}(\log \log N)^{2/3})$ . In 2006, it was known in practice how to factor an integer with 100 digits in a couple of hours, and by using many computers over many months, how to factor an integer with 150 digits. But we still cannot factor, over the course of a human lifetime, an RSA number, with say 300 digits. A surprising fact is that the complexity of various algorithms (proven probabilistically or heuristically) tends to take the form of a function (see [48]):

$$L(b, N) := \exp(C(\log N)^b(\log \log N)^{1-b}).$$

The case  $b = 0$ , in other words  $(\log N)^C$ , corresponds to polynomial algorithms, the case  $b = 1$ , in other words  $N^C$ , corresponds to exponential algorithms and the cases  $0 < b < 1$  correspond to subexponential algorithms; the two algorithms cited above have a complexity estimated at  $L(1/2, N)$  and  $L(1/3, N)$ .

We are not going to present the most powerful algorithms right away, since they use tools which surpass the level of this chapter; the algorithms which use elliptic curves and the number field sieve are presented in Appendix A, which is about factorization. For the moment, we will settle for describing an algorithm which improves on the naive algorithm by providing an even more efficient one.

From now on, we use the convention that the letter  $p$  is reserved for a factor of  $N$ .

**Pollard’s  $\rho$  algorithm.** We proceed as follows. We choose  $a_0$  between 1 and  $N$  and we compute the sequence given by  $a_{i+1} = f(a_i)$ , where  $f(a) := a^2 + 1 \bmod N$ . We then choose  $k$  “big enough, but not too big” and we calculate  $\gcd(a_{2k} - a_k, N)$ , hoping that it is nontrivial; if that is the case, we have found a factorization, if not, we try again with larger  $k$ . We will explain below why, at least statistically, there exists  $k$  of size  $O(\sqrt{p})$ , where  $p$  divides  $\gcd(a_{2k} - a_k, N)$ . Assuming that, we see that the average complexity of the algorithm is  $O(\sqrt{p})$ , thus  $O(\sqrt[4]{N})$ .

The analysis of the complexity is based on the hypothesis that the sequence  $a_i$  modulo  $p$  is sufficiently “random”, which has been satisfactorily confirmed in practice. Now, the probability that  $r$  numbers modulo  $p$  chosen “at random” are all distinct is<sup>6</sup>

$$P_r = \left(1 - \frac{1}{p}\right) \left(1 - \frac{2}{p}\right) \cdots \left(1 - \frac{r-1}{p}\right) \leq \exp\left(-\frac{r(r-1)}{2p}\right).$$

If we take  $r$  on the order of  $\sqrt{p}$ , say  $r \geq 2\sqrt{p}$ , the probability that two

---

<sup>6</sup>Example. If  $n \geq 23$ , the probability that, among  $n$  people, two have the same birthday is greater than  $1/2$ .

of the numbers are equal (modulo  $p$ ) will be  $> 1/2$ , thus we have a good chance to have two indices  $i < j < r$  such that  $a_i \equiv a_j \pmod{p}$ . Considering the construction that follows, we would have  $a_{i+m} \equiv a_{j+m} \pmod{p}$  for every  $m \geq 0$ , and in particular, by taking  $m = j - 2i$  and  $k = j - i$ , we would have  $a_k \equiv a_{2k} \pmod{p}$  (see [22] for more details).

**“Difference of squares” algorithm.** The second algorithm, that we will only sketch, is based on the fact that the number of elements  $a \in (\mathbf{Z}/N\mathbf{Z})^*$  such that  $a^2 = 1$  is at least equal to 4 if  $N$  has at least two distinct prime factors. If we knew how to compute a square root in  $(\mathbf{Z}/N\mathbf{Z})^*$ , say  $\mathcal{A}(x)$ , with a fast algorithm  $\mathcal{A}$ , then we could factor  $N$  like this: take  $a$  at random and calculate  $b = \mathcal{A}(a^2)$ . Then we of course have that  $a^2 \equiv b^2 \pmod{N}$ , or even that  $N$  divides  $(a + b)(a - b)$ . Now, there is (at least) a one in two chance that  $\pm a \pmod{N}$  is not the square root calculated by  $\mathcal{A}$  and, in this case, the calculation of  $\gcd(N, a + b)$  or of  $\gcd(N, a - b)$  would give us a factorization. Unfortunately, or luckily, we do not know of any fast algorithm  $\mathcal{A}$  (it is even possible that one does not exist). One extension of this idea is the following: instead of directly looking for an equality  $a^2 \equiv b^2 \pmod{N}$ , we try to construct one. In order to do this, we randomly take  $a$  close to  $\sqrt{N}$ , we reduce  $a^2$  modulo  $N$  (taking care to take the representative in  $[-N/2, N/2]$ ) and we try to factor it with small prime numbers. In this way, we get a family of congruences  $a_j^2 \equiv \prod_{p \in S} p^{n_{p,j}}$ . We therefore look for a combination of these numbers which provides an equality of the type  $\prod_i a_i^2 \equiv \prod_j b_j^2 \pmod{N}$  (this is a linear algebra problem over  $\mathbf{F}_2$ ). This idea, presented very vaguely here, is expanded on in more detail in Appendix A, when we describe the number field sieve algorithm. Property quantified, this algorithm has an average (heuristic) complexity on the order of  $L(1/2, N)$ —which is already remarkable, even if it is insufficient for very large numbers.

**Examples of precautions to take when choosing  $p$  and  $q$  for the RSA method.** We will only give some very elementary indications, since the question is fairly complex, and in fact largely open.

1) The absolute value,  $|p - q|$ , must be large. We can see why by writing  $q = p + \delta$  where  $\delta$  is much smaller than  $p$ . Since  $N = pq$ , then  $\sqrt{N} = p\sqrt{1 + \delta/p} \sim p + \delta/2$  and we could find  $p$  with the “naive” algorithm in  $O(\delta)$  steps!

2) It must be that  $p - 1$  (resp.  $q - 1$ ) are not too *smooth*, in other words, cannot be factored too quickly, for example the product of small prime numbers. To see why this is true, choose  $C > 0$ , and let  $p_1, \dots, p_k$  be the prime numbers smaller than  $C$ ; the set  $S := \{s = p_1^{m_1} \cdots p_k^{m_k} \mid s \leq N\}$  has cardinality  $O((\log N)^k)$ , and we can therefore calculate  $\gcd(a^s - 1, N)$  for some values of  $a$  and  $s \in S$  in polynomial time. If  $p - 1 \in S$  (in other words

if  $p - 1$  only has prime factors  $\leq C$ ), then we have a very good chance of being able to factor  $N$ .

3) A less obvious constraint is that it must be that the “secret” exponent  $e$  is not too small. It is clear that if  $e = O(\log N)$  for example, then by trying  $O(\log N)$  times, we will find  $e$ , but in fact it can be shown that you must avoid having  $e \ll N^{1/4}$  (see Exercise 3-6.12 of Chap. IV).

These relatively trivial remarks could cast doubt the security of the RSA system (see [17] for a more precise description of the catalogued attacks on the RSA system). However, theoretical support for it is provided by the following considerations. Let us call  $P$  the class of problems for which there exists a polynomial algorithm (for example the problem of deciding whether a number is prime is in  $P$ , by Agrawal-Kayal-Saxena). We can define a class  $NP$ , a priori much larger than  $P$ , which is the class of problems for which there exists a polynomial verification (for example, the problem of factorization of a number is clearly in  $NP$ , since if we are given a factorization, we can verify it in polynomial time). However, the factorization problem has a subexponential solution. The security of the RSA system rests, from a theoretical point of view, on the hypothesis that the factorization problem is not in  $P$ . In fact, it is a special case of a large problem in complexity theory<sup>7</sup>:

Is it true that  $P \neq NP$ ?

## 6. Error-Correcting Codes

*We give a glimpse of another industrial application of algebra and arithmetic: the construction of “error-correcting codes”, which can, to a certain degree, reconstruct a message if its transmission was slightly defective. This technique is for example needed to produce CD readers, to transmit images by space probes, etc. If this introduction leaves you hungry to learn more, I recommend Demazure’s book, Cours d’algèbre [3].*

### 6.1. Generalities about Error-Correcting Codes

In order to transmit information, we assume that we are using a finite alphabet  $\mathcal{Q}$ , containing  $q$  symbols or letters and that we are sending words of a fixed length  $n$ ; a word is therefore an element of  $\mathcal{Q}^n$ . We can think of binary language, i.e.,  $\mathcal{Q} := \{0, 1\}$ , or of genetic codes, for example  $\mathcal{Q} := \{A, C, G, U\}$  (the bases found in RNA are A for adenine, C for cytosine, G for guanine and U for uracil). We will most often take the example of

---

<sup>7</sup>This problem  $P \neq NP$  is one of the seven problems, for the solution of which a million dollars is offered by the Clay Mathematics Institute.

$\mathcal{Q} := \mathbf{F}_q$ , which has the disadvantage of limiting the possible values of  $q$  but the advantage of providing a richer structure.

The set of words  $\mathcal{Q}^n$  can be endowed with a *Hamming distance*, defined as follows. If  $x = (x_1, \dots, x_n) \in \mathcal{Q}^n$  and  $x' = (x'_1, \dots, x'_n) \in \mathcal{Q}^n$ , then

$$d(x, x') := \text{card}\{i \in [1, n] \mid x_i \neq x'_i\}.$$

It can easily be checked that is in fact a distance.

A *code* is a subset  $\mathcal{C} \subset \mathcal{Q}^n$  containing at least two distinct elements in  $\mathcal{Q}^n$ ; we define the *distance* of a code as

$$d(\mathcal{C}) := \min_{x \neq x' \in \mathcal{C}} d(x, x').$$

Once we have chosen a code  $\mathcal{C}$ , the principle consists of only sending those messages which belong to  $\mathcal{C}$ . If we know that at most  $d(\mathcal{C}) - 1$  transmission errors have been committed, then using the error-correcting code will enable us to establish the existence of one or more errors. Furthermore, if  $t$  errors have been committed during the transition of a word and if  $2t + 1 \leq d(\mathcal{C})$ , we see that there exists one single word in  $\mathcal{C}$  located at a distance  $\leq t$  from the received word. In conclusion, the code allows us to correct  $t$  errors and we say that it is *t-correcting*. If we denote by  $d = d(\mathcal{C})$  the distance of the code and  $t = t(\mathcal{C})$  the number of errors that are systematically corrected by the code, we easily see that relationship between the two is given by  $t = \left\lfloor \frac{d-1}{2} \right\rfloor$  and conversely  $d = 2t + 1$  or  $2t + 2$ . Except for some examples, we leave aside the question of *decoding*, which is essentially the study of algorithms which allow you to find the word of the code located at a minimal distance from a given word (it should be noted that you cannot in general guarantee the uniqueness of this word except under certain conditions). One of the properties required of a code is obviously that it corrects or finds the most possible errors (we could also insist that the decoding be the simplest possible). An intuitively obvious requirement is that it uses the least amount of space; we could formalize this idea by introducing the *code rate*  $t/n$ , and the *information rate* that we define as the ratio  $\log \text{card}(\mathcal{C}) / n \log q$ . Information theory, developed by Shannon (see the founding article [67]), says that if we are willing to send longer and longer messages (i.e., to let  $n$  be very large), then there exist codes as safe we want them to be, with an information rate close to 1. Shannon's theorem is however an existence theorem, it does not specify how to construct such codes.

We are actually going to exclusively concentrate on *linear codes*, where the alphabet is (in bijection with)  $\mathbf{F}_q$ , the space of words is (in bijection with) the vector space  $(\mathbf{F}_q)^n$  and  $\mathcal{C}$  is a subspace. In the case of  $q = 2$ , we are

talking about binary codes, in the case  $q = 3$ , we are talking about ternary codes, etc.

The most important parameters of a linear code are the cardinality of the alphabet  $q = \text{card } \mathcal{Q}$ , its *length* say  $n$ , its *dimension* say  $k := \dim \mathcal{C}$ , its distance  $d(\mathcal{C})$ , its code rate and its information rate  $k/n$ .

Remark. Let  $\mathcal{C} \subset \mathbf{F}_q^n$  be a linear code. We define the *weight* of an element  $w(x)$  as the number of non-zero components of  $x$ . We can easily see that

$$d(\mathcal{C}) = \min_{0 \neq x \in \mathcal{C}} d(0, x) = \min_{0 \neq x \in \mathcal{C}} w(x).$$

**6.1.1. Examples.** 1) The most basic example of a code is the use of a *parity bit*: in order to transmit a word  $x = (x_1, \dots, x_{n-1}) \in (\mathbf{F}_2)^{n-1}$ , we send  $\bar{x} = (x_1, \dots, x_{n-1}, x_1 + \dots + x_{n-1}) \in (\mathbf{F}_2)^n$ . To see if the received message  $x' = (x_1, \dots, x_n)$  is correct, we check whether  $x_n = x_1 + \dots + x_{n-1}$ . This code has length  $n$  and dimension  $n - 1$ . It allows us to find an error but not to correct it. Its distance is 2.

2) Hamming code. Take the set of words with seven binary digits,  $q = 2$ ,  $n = 7$ , and let  $\mathcal{C}$  be the code with basis

$$e_0 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad e_1 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad e_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad e_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

The coding principle is simple: in order to transmit a message  $m = (m_0, m_1, m_2, m_3)$ , we transmit  $x = m_0 e_0 + m_1 e_1 + m_2 e_2 + m_3 e_3$ . For this simple example, we will explain the decoding *under the hypothesis that at most one error was committed*. Equations of the vector subspace  $\mathcal{C}$  are given by

$$L(x) = (x_0 + x_3 + x_5 + x_6, x_1 + x_3 + x_4 + x_6, x_2 + x_4 + x_5 + x_6) = 0.$$

For each vector  $e$  of weight 1, we then calculate the triplet  $L(e)$ . From this, we obtain the following algorithm of correction and decoding. After having received the message  $x = (x_0, \dots, x_6)$ , we check whether  $L(x) = 0$ . If  $L(x) = 0$ , the message is correct, if  $L(x) = (1, 0, 0)$ , then  $x_0$  must be corrected, if  $L(x) = (0, 1, 0)$ , then  $x_1$  must be corrected and if  $L(x) = (1, 0, 1)$ , then  $x_5$  must be corrected. Finally, if  $L(x) = (1, 1, 1)$ , then  $x_6$  must be corrected. Thus we have  $m = (x_0, x_0 + x_1, x_5, x_6)$ .

We denote by  $T(x_1, \dots, x_7) := (x_7, x_1, \dots, x_6)$  the “shift”, so we have that  $T(e_0) = e_1$ ,  $T(e_1) = e_2$ ,  $T(e_2) = e_3$  and  $T(e_3) = e_0 + e_1 + e_2$ . Thus

$T(\mathcal{C}) = \mathcal{C}$  ( $\mathcal{C}$  is then called cyclic). It is easy to see that each non-zero vector in  $\mathcal{C}$  has at least three non-zero coordinates, and therefore  $d(\mathcal{C}) = 3$ . Therefore, this code is 1-correcting and allows us to identify two errors but not to correct them.

An amusing example. The previous code suggests that it is possible to recover an element of  $\mathbf{F}_2^4$  (or say an integer between 0 and 15) starting with an element of  $\mathbf{F}_2^7$  (or say seven yes/no pieces of information) if at most one error has been committed (granted at most one of the bits of information is false). One version of this is the seven following questions which allow us to determine an integer  $N$  between 0 and 15.

- 1) Is the integer  $N \geq 8$ ?
- 2) Is the integer  $N$  in the set  $\{4, 5, 6, 7, 12, 13, 14, 15\}$ ?
- 3) Is the integer  $N$  in the set  $\{2, 3, 6, 7, 10, 11, 14, 15\}$ ?
- 4) Is the integer  $N$  odd?
- 5) Is the integer  $N$  in the set  $\{1, 2, 4, 7, 9, 10, 12, 15\}$ ?
- 6) Is the integer  $N$  in the set  $\{1, 2, 5, 6, 8, 11, 12, 15\}$ ?
- 7) Is the integer  $N$  in the set  $\{1, 3, 4, 6, 8, 10, 13, 15\}$ ?

We leave as an exercise the justification of the following algorithm. We denote the answers to the above questions by  $m = (m_1, \dots, m_7)$  ( $m_i = 1$  if the  $i$ th answer is yes,  $m_i = 0$  if not), and we compute  $a_1 = m_4 + m_5 + m_6 + m_7$ ,  $a_2 = m_2 + m_3 + m_6 + m_7$  and  $a_3 = m_1 + m_3 + m_5 + m_7$ . If  $a_1 = a_2 = a_3 = 0$ , we conclude that there is not an error, if not we change the  $r$ th answer  $m_r$  into  $r = \overline{a_1 a_2 a_3}$  (binary numeral notation), and the number we are looking for is therefore written

$$N = \overline{m_1 m_2 m_3 m_4}.$$

We will now show how to characterize and construct codes and how to deduce new codes from the given ones by using elementary linear algebra. We denote by  $n$  the length of the codes and by  $k$  their dimension, unless specified otherwise.

**6.1.2. Definition.** A *generator* matrix of a code  $\mathcal{C}$  is a matrix whose rows form a basis of  $\mathcal{C}$ . (It is therefore a matrix of rank  $k$  having  $k$  rows and  $n$  columns.) A *parity-check* matrix of a code  $\mathcal{C}$  is a matrix whose rows form a basis for the linear forms which are zero over  $\mathcal{C}$ . (It is therefore a matrix of rank  $n - k$  having  $n - k$  rows and  $n$  columns.)

**6.1.3. Remarks.** Being given a generator matrix is of course equivalent to being given a basis of the vector space  $\mathcal{C}$ , and given a parity-check matrix is of course equivalent to being given a basis of linear equations which define  $\mathcal{C}$  in  $\mathbf{F}_q^n$ . If  $A$  is a generator matrix and  $B$  a parity-check

matrix, we easily see that  $A^t B = 0$ , or also  $B^t A = 0$ . Moreover, we can recognize the distance of the code as the smallest number  $d$  such that there exist  $d$  dependent column vectors in  $B$ .

Assume that we are given a code  $\mathcal{C}$  with parity-check matrix  $B$  and assume that the code is 1-correcting. We show you how to decode a received message,  $x'$ , which is different in at least one coordinate from the sent message,  $x$ . First of all, if we denote the error committed by  $\epsilon = x' - x$ , we see that  $B(x') = B(\epsilon)$ . We will therefore compute  $B(x')$ ; if this is non-zero, then no error has been committed, if not, we compute the images of the vectors  $e_i$  in the canonical basis  $f_i = B(e_i)$ . If only one error has been committed, we find a unique  $i$  such that  $B(x')$  is proportional to  $f_i$ , say  $B(x') = a_i f_i$ , and therefore  $\epsilon = a_i e_i$  and  $x = x' - a_i e_i$ .

If  $\mathcal{C}$  is a code of length  $n$  over the field  $\mathbf{F} = \mathbf{F}_q$ , we can associate to it the following codes.

- i) *Shortened code*. Let  $d(\mathcal{C}) \leq \ell \leq n$ . We set  $\mathcal{C}^{(\ell)} := \{x \in \mathbf{F}_q^\ell \mid (x; 0, \dots, 0) \in \mathcal{C}\}$ . It is a code of length  $\ell$ , and we easily see that  $d(\mathcal{C}^{(\ell)}) \geq d(\mathcal{C})$ .
- ii) *Extended code*. We can create the analogue of the “parity bit” by constructing  $\bar{\mathcal{C}} := \{(x_1, \dots, x_{n+1}) \in \mathbf{F}_q^{n+1} \mid (x_1, \dots, x_n) \in \mathcal{C} \text{ and } x_1 + \dots + x_n + x_{n+1} = 0\}$ . We can easily see that  $d(\mathcal{C}) \leq d(\bar{\mathcal{C}}) \leq d(\mathcal{C}) + 1$ . One variation is the *even subcode* defined as  $\mathcal{C}' = \{x \in \mathcal{C} \mid x_1 + \dots + x_n = 0\}$ . We have  $d(\mathcal{C}) \leq d(\mathcal{C}')$ .
- iii) *Dual code*. We define the scalar product  $\langle x, y \rangle := x_1 y_1 + \dots + x_n y_n$ , and we set  $\mathcal{C}^* := \{x' \in \mathbf{F}_q^n \mid \forall x \in \mathcal{C}, \langle x, x' \rangle = 0\}$ . We have that  $\dim \mathcal{C}^* = n - \dim \mathcal{C}$ . An interesting category of binary codes is that of *self-dual* codes, i.e., such that  $\mathcal{C}^* = \mathcal{C}$ ; such codes have dimension  $n/2$ , and the weight of an element is even since  $w(x) \equiv \langle x, x \rangle \pmod{2}$ .

As an exercise, you could try to figure out how to construct a parity-check (or generator) matrix of each of these codes, starting with the parity-check (or generator) matrix of the original code.

**6.1.4. Lemma.** *Let  $\mathcal{C}$  be a code of dimension  $k$  and of length  $n$  over  $\mathbf{F}_q$ . The following inequalities hold:*

- i)  $d(\mathcal{C}) \leq n + 1 - k$  ;
- ii) if  $\mathcal{C}$  is  $t$ -correcting  $1 + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t \leq q^{n-k}$ .

*Proof.* i) The vectors of the form  $(x_1, \dots, x_{n+1-k}, 0, \dots, 0)$  form a vector subspace  $\mathcal{D}$  of  $(\mathbf{F}_q)^n$ . Since  $\dim \mathcal{D} + \dim \mathcal{C} = n+1$ , we see that  $\mathcal{D} \cap \mathcal{C} \neq \{0\}$ , hence the existence of a non-zero vector of  $\mathcal{C}$  of weight  $\leq n + 1 - k$ . For



ii), we can observe that for every  $x \in \mathbf{F}_q^n$  and  $0 \leq t \leq n$ ,

$$\text{card}(B(x, t)) = 1 + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t.$$

If the code is  $t$ -correcting, the balls  $B(x, t)$  with center  $x \in \mathcal{C}$  are disjoint and thus

$$\text{card}(\cup_{x \in \mathcal{C}} B(x, t)) = q^k \text{card}(B(0, t)) \leq q^n. \quad \square$$

**6.1.5. Definition.** A code such that  $d(\mathcal{C}) = n + 1 - k$  is called MDS *maximal distance separable*. A  $t$ -correcting code such that  $\mathcal{C} = \cup_{x \in \mathcal{C}} B(x, t)$  (forcibly a disjoint union) is called *perfect  $t$ -correcting*.

The Hamming code of length 7 studied in the examples is perfect 1-correcting since, in this case, we can show that  $\text{card } B(x, 1) = 1 + 7 = 8$  and  $8 \text{ card } \mathcal{C} = 2^7$ . We could also notice that this code is not MDS, because  $d(\mathcal{C}) = 3 < 4 = n - k + 1$ .

## 6.2. Linear Cyclic Codes

*We will explicitly describe an interesting class of codes which in particular contains some of the classical codes, such as that of Hamming, Reed-Solomon and Golay and which will lead us into the study of cyclotomic polynomials.*

**6.2.1. Definition.** A linear cyclic code is a linear code,  $\mathcal{C}$ , of length  $n$ , which is stable under the transformation  $T(a_0, a_1, \dots, a_{n-1}) = (a_{n-1}, a_0, \dots, a_{n-2})$ .

We can give a nice algebraic characterization of cyclic codes by introducing the natural isomorphism of vector spaces  $\mathbf{F}_q^n \cong \mathbf{F}_q[X]_n \cong \mathbf{F}_q[X]/Q\mathbf{F}_q[X]$ , where  $\mathbf{F}_q[X]_n$  represents the polynomials of degree  $< n$  and where  $Q$  is a polynomial of degree  $n$ . Since the characteristic (or minimal) polynomial of the endomorphism  $T$  is  $Q = X^n - 1$ , we therefore choose this value. Hence we denote by  $\psi : \mathbf{F}_q^n \rightarrow \mathbf{F}_q[X]_n \cong \mathbf{F}_q[X]/(X^n - 1)$  defined as  $\psi(a_0, a_1, \dots, a_{n-1}) \mapsto a_0 + a_1X + \cdots + a_{n-1}X^{n-1} \bmod (X^n - 1)$ . We immediately see that

$$\psi \circ T(a_0, a_1, \dots, a_{n-1}) = X(a_0 + a_1X + \cdots + a_{n-1}X^{n-1}) \bmod (X^n - 1).$$

Thus a vector subspace  $\mathcal{C} \subset \mathbf{F}_q^n$  is stable under  $T$  if and only if its image under  $\psi$  is stable under multiplication by  $X$ . We should point out that an  $\mathbf{F}_q$  vector subspace of  $\mathbf{F}_q[X]/(X^n - 1)$  which is stable under multiplication by  $X$  is nothing other than an *ideal* of  $\mathbf{F}_q[X]/(X^n - 1)$ . Finally, the ideals of  $\mathbf{F}_q[X]/(X^n - 1)$  correspond to the ideals of  $\mathbf{F}_q[X]$  which contain the

polynomial  $X^n - 1$  and therefore are of the form  $P\mathbf{F}_q[X]$  where  $P$  divides  $X^n - 1$ . We summarize this discussion in the following theorem.

**6.2.2. Theorem.** *Let  $K := \mathbf{F}_q$  and let  $\mathcal{C}$  be a cyclic code of length  $n$ . We identify  $K^n$  with  $K[X]/(X^n - 1)$  via  $(a_0, a_1, \dots, a_{n-1}) \mapsto a_0 + a_1X + \dots + a_{n-1}X^{n-1}$ . There exist natural bijections between the following objects:*

- i) a cyclic code of length  $n$ ;*
- ii) an ideal  $K[X]/(X^n - 1)$ ;*
- iii) a monic polynomial which divides  $X^n - 1$  in  $K[X]$ .*

*One of the bijections associates  $P$ , which divides  $X^n - 1$ , to the ideal  $\mathcal{C}$  of  $K[X]/(X^n - 1)$  generated by its class modulo  $X^n - 1$ , and another associates an ideal of  $K[X]/(X^n - 1)$  to the vector subspace corresponding to  $\mathcal{C}$  of  $K^n$ . Furthermore,  $\dim \mathcal{C} = n - \deg(P)$ .*

This leads to the following problem: how to decompose the polynomial  $X^n - 1$  in  $\mathbf{F}_q[X]$ ?

It is of course better to start with a decomposition in  $\mathbf{Z}[X]$  (or  $\mathbf{Q}[X]$ ), which is provided by *cyclotomic polynomials*. In order to define these, we denote by  $\mu_n = \{\zeta \in \mathbf{C} \mid \zeta^n = 1\}$  the group of  $n$ th roots of unity and  $\mu_n^*$  the subset of  $n$ th primitive roots of unity, and hence  $\text{card } \mu_n = n$  and  $\text{card } \mu_n^* = \phi(n)$ .

We will need Gauss's lemma.

**6.2.3. Lemma.** *If  $P = p_0 + p_1X + \dots + p_dX^d \in \mathbf{Z}[X]$  is a non-zero polynomial, we define its content as  $c(P) := \gcd(p_0, \dots, p_d)$ . We therefore have that*

$$c(PQ) = c(P)c(Q).$$

*Proof.* By factoring  $P = c(P)P^*$  and  $Q = c(Q)Q^*$ , we see that  $c(PQ) = c(P)c(Q)c(P^*Q^*)$ . So we have reduced the proof to showing that if  $P$  and  $Q$  are primitive (i.e.,  $c(P) = c(Q) = 1$ ), then  $c(PQ) = 1$ . If  $p$  is a prime number, we denote by  $\bar{P}$  the image of  $P$  in  $\mathbf{F}_p[X]$ . We have that  $\bar{P} \neq 0$  and  $\bar{Q} \neq 0$ , thus  $\bar{P} \cdot \bar{Q} = \overline{PQ} \neq 0$  because  $\mathbf{F}_p[X]$  is integral. So no  $p$  divides  $c(PQ)$ , which implies that it is invertible.  $\square$

**6.2.4. Corollary.** *Let  $P \in \mathbf{Z}[X]$ . Suppose that there exist  $Q, R \in \mathbf{Q}[X]$  such that  $P = QR$ . Then there exists  $\lambda \in \mathbf{Q}^*$  such that  $\lambda Q$  and  $\lambda^{-1}R$  have integer coefficients.*

*Proof.* We can write  $Q = \frac{a}{b}Q_1$  (resp.  $R = \frac{c}{d}R_1$ ), where  $a, b, c, d$  are integers and where  $Q_1$  and  $R_1$  are primitive polynomials with integer coefficients. We can deduce from this that  $bdP = acQ_1R_1$  and, since the

equality is in  $\mathbf{Z}[X]$ , we can deduce, using Gauss's lemma, that  $bd \mid c(P) = ac$  and, in particular, that  $bd$  divides  $ac$ . Thus  $P = c(P)Q_1R_1$ .  $\square$

**6.2.5. Corollary.** *If  $\alpha \in \mathbf{C}$  is a root of a monic polynomial with integer coefficients, then the minimal (monic) polynomial of  $\alpha$  has integer coefficients.*

*Proof.* Let  $P$ , a priori in  $\mathbf{Q}[X]$ , be the minimal polynomial of  $\alpha$  and let  $Q$  be monic with integer coefficients such that  $Q(\alpha) = 0$ . Then  $Q = PR$ , where  $R$  is in  $\mathbf{Q}[X]$ . Gauss's lemma says that there exists  $\lambda \in \mathbf{Q}^*$  such that  $R_0 = \lambda R$  and  $P_0 = \lambda^{-1}P$  have integer coefficients. By observing that  $Q = P_0R_0$ , it follows that the leading coefficient of  $P_0$  is invertible, and hence  $P = \pm P_0$  has integer coefficients.  $\square$

**6.2.6. Definition.** The  $n$ th cyclotomic polynomial, denoted  $\Phi_n$ , is defined as

$$\Phi_n(X) := \prod_{\zeta \in \mu_n^*} (X - \zeta).$$

These polynomials, a priori with complex coefficients, in fact have integer coefficients and moreover provide a decomposition of  $X^n - 1$  into irreducible factors, as shown in the following theorem.

**6.2.7. Theorem.** *The polynomials  $\Phi_n$  have the following properties.*

- i)  $\Phi_n \in \mathbf{Z}[X]$  and  $\deg \Phi_n = \phi(n)$ .
- ii)  $X^n - 1 = \prod_{d \mid n} \Phi_d(X)$ .
- iii) *The polynomials  $\Phi_n$  are irreducible in  $\mathbf{Z}[X]$  and in  $\mathbf{Q}[X]$ .*

*Proof.* With the given definition,  $\Phi_n \in \mathbf{C}[X]$ . Formula ii) is clear, as well as the fact that  $\deg(\Phi_n) = \phi(n)$ ; however it is less clear that in fact  $\Phi_n \in \mathbf{Z}[X]$  and that  $\Phi_n$  is irreducible in  $\mathbf{Q}[X]$  (or  $\mathbf{Z}[X]$ ). We shall start by showing that the coefficients of  $\Phi_n$  are integers. It is clear that  $\Phi_1(X) = X - 1 \in \mathbf{Z}[X]$ , and formula ii) leads us to try induction on  $n$ . The polynomial  $B := \prod_{d \mid n, d \neq n} \Phi_d(X)$  is monic and, by applying induction, has integer coefficients. We can therefore carry out the division algorithm in  $\mathbf{Z}[X]$ , and obtain  $X^n - 1 = BQ + R$ . Formula ii) then guarantees that  $B$  divides  $R$  (in  $\mathbf{Q}[X]$ ), so  $R = 0$  and  $Q = \Phi_n$ . We will now show that  $\Phi_n$  is irreducible in  $\mathbf{Z}[X]$ . Let  $\zeta$  be a primitive  $n$ th root of unity and  $P$  its minimal polynomial over  $\mathbf{Q}$ . We therefore need to show that  $P = \Phi_n$ . First, observe that  $P \in \mathbf{Z}[X]$ . Then choose a prime number  $p$  which does not divide  $n$ , so  $\zeta^p$  is still an  $n$ th primitive root of unity. Let  $Q$  be its minimal polynomial, which is also in  $\mathbf{Z}[X]$ . If  $P$  and  $Q$  were distinct, the product  $PQ$  would

divide  $\Phi_n$ . But since  $Q(\zeta^p) = 0$ , we see that  $\zeta$  is a root of  $Q(X^p)$  and thus  $Q(X^p) = P(X)R(X)$ , for some  $R \in \mathbf{Z}[X]$ . By reducing the coefficients modulo  $p$ , we have

$$\bar{Q}(X^p) = \bar{Q}(X)^p = \bar{P}(X)\bar{R}(X),$$

and so  $\bar{P}(X)$  divides  $\bar{Q}(X)^p$  in  $(\mathbf{Z}/p\mathbf{Z})[X]$ . Moreover, the factors of  $X^n - 1$ , and hence of  $\bar{P}(X)$ , are simple in  $(\mathbf{Z}/p\mathbf{Z})[X]$  (the derivative of  $X^n - 1$  is  $nX^{n-1}$ , and we made a point of choosing  $p$  so that it does not divide  $n$ ): the polynomial  $\bar{P}(X)$  in fact divides  $\bar{Q}(X)$ . But then,  $\bar{P}(X)^2$  divides  $\bar{\Phi}_n(X)$  in  $(\mathbf{Z}/p\mathbf{Z})[X]$ , which contradicts the fact that the factors of  $\bar{\Phi}_n(X)$  are simple. To summarize, we have established that if  $p$  is a prime number which does not divide  $n$ , the minimal polynomial of  $\zeta$  kills  $\zeta^p$ . We easily deduce from this that if  $m$  is relatively prime to  $n$ , then  $P(\zeta^m) = 0$ . Thus  $\deg(P) \geq \phi(n)$  and since  $P$  divides  $\Phi_n$ , we have that  $P = \Phi_n$ , and it is therefore irreducible.  $\square$

Since  $\Phi_n$  has integer coefficients, we can reduce its coefficients modulo  $p$  and consider it as a polynomial in  $\mathbf{F}_p[X]$  (or in  $\mathbf{F}_q[X]$  with  $q = p^f$ ).

**6.2.8. Theorem.** *The decomposition into irreducible factors of the polynomial  $\Phi_n \in \mathbf{F}_q[X]$  (with  $q = p^f$ ) depends on whether  $n$  modulo  $p$  is zero or not.*

- i) *If  $n = p^s m$  where  $p \nmid m$ , we have  $\Phi_n(X) = \Phi_m(X)^{p^s - p^{s-1}}$ .*
- ii) *If  $\gcd(n, q) = 1$  and if  $r$  is the order of  $q \bmod n$  in  $(\mathbf{Z}/n\mathbf{Z})^*$ , then  $\Phi_n$  can be decomposed into the product of  $\phi(n)/r$  distinct irreducible factors of degree  $r$ .*

*Proof.* Assume first that  $n = p^r m$ . By Fermat's little theorem and the formulas from Exercise 2-7.12, it follows that  $\Phi_m(X)^p \equiv \Phi_m(X^p) = \Phi_{mp}(X)\Phi_m(X)$ , hence  $\Phi_{mp}(X) \equiv \Phi_m(X)^{p-1}$ , and subsequently that

$$\Phi_{mp^r}(X) = \Phi_{mp} \left( X^{p^{r-1}} \right) \equiv \Phi_{mp}(X)^{p^{r-1}} \equiv \Phi_m(X)^{p^{r-1}(p-1)},$$

which proves the first assertion. From now on, suppose that  $p$  is relatively prime to  $n$ . Let  $\beta$  be an  $n$ th primitive root in an extension of  $\mathbf{F}_q$ . Every factor of  $\Phi_n$  can be written as  $Q = \prod_{i \in I} (X - \beta^i)$ , with  $I \subset (\mathbf{Z}/n\mathbf{Z})^*$ . The polynomial  $Q$  has coefficients in  $\mathbf{F}_q$  if and only if

$$Q(X)^q = Q(X^q). \quad (*)$$

In fact,  $\left( \sum_j a_j X^j \right)^q = \sum_j (a_j)^q X^{qj}$  and  $a \in \mathbf{F}_q$  if and only if  $a^q = a$ . Thus the polynomial  $Q$  has coefficients in  $\mathbf{F}_q$  if and only if

$$\prod_{i \in I} (X^q - \beta^{iq}) = \prod_{i \in I} (X - \beta^i)^q = \prod_{i \in I} (X^q - \beta^i),$$

or even if and only if  $I$  is stable under multiplication by  $q$  (in  $(\mathbf{Z}/n\mathbf{Z})^*$ ). The smallest stable subset is clearly of the form  $I := \{i, iq, iq^2, \dots, iq^{r-1}\}$ . Also, the irreducible factors of  $\Phi_n(X)$  in  $\mathbf{F}_q[X]$  are of the form

$$Q = \prod_{s=0}^{r-1} (X - \beta^{iq^s}),$$

and, in particular, all have degree  $r$ . □

**6.2.9. Examples.** 1) Take  $n = 11$  and  $q = 3$ ; we see that the order of  $3 \bmod 11$  is equal to 5. Thus  $X^{11} - 1 = (X - 1)\Phi_{11}(X)$  in  $\mathbf{Z}[X]$  and  $\Phi_{11} = P_1 P_2 \in \mathbf{F}_3[X]$ , where  $\deg(P_i) = 5$ . We can check that, in  $\mathbf{F}_3[X]$ ,

$$X^{11} - 1 = (X - 1)(X^5 - X^3 + X^2 - X - 1)(X^5 + X^4 - X^3 + X^2 - 1).$$

2) Take  $n = 23$  and  $q = 2$ ; we see that the order of  $2 \bmod 23$  is equal to 11. Thus  $X^{23} - 1 = (X - 1)\Phi_{23}(X)$  in  $\mathbf{Z}[X]$  and  $\Phi_{23} = P_1 P_2 \in \mathbf{F}_2[X]$ , with  $\deg(P_i) = 11$ . We can check that, in  $\mathbf{F}_2[X]$ ,

$$\begin{aligned} X^{23} - 1 &= (X - 1)(X^{11} + X^{10} + X^6 + X^5 + X^4 + X^2 + 1) \\ &\quad \times (X^{11} + X^9 + X^7 + X^6 + X^5 + X + 1). \end{aligned}$$

3) Take  $n = 15$  and  $q = 2$ ; thus  $X^{15} - 1 = (X - 1)\Phi_3(X)\Phi_5(X)\Phi_{15}(X)$  in  $\mathbf{Z}[X]$ , with  $\Phi_{15} = X^8 - X^7 + X^5 - X^4 + X^3 - X + 1$ . The order of  $2 \bmod 3$  is equal to 2, the order of  $2 \bmod 5$  is equal to 4 and the order of  $2 \bmod 15$  is equal to 4. The polynomials  $\Phi_3 = X^2 + X + 1$  and  $\Phi_5 = X^4 + X^3 + X^2 + X + 1$  are therefore irreducible in  $\mathbf{F}_2[X]$ , and  $\Phi_{15} = P_1 P_2 \in \mathbf{F}_2[X]$ , where  $\deg(P_i) = 4$ . We can check that, in  $\mathbf{F}_2[X]$ ,

$$X^{15} - 1 = (X - 1)(X^2 + X + 1)(X^4 + X^3 + X^2 + X + 1)(X^4 + X^3 + 1)(X^4 + X + 1).$$

4) More generally, if  $\gcd(q, n) = 1$ , a cyclic code of length  $n$  corresponds, by Theorem 2-6.2.2, to a subset  $I \subset \mathbf{Z}/n\mathbf{Z}$ , which is stable under multiplication by  $q$ . More explicitly, the associated code is the ideal of  $\mathbf{F}_q[X]/(X^n - 1)$  generated by the polynomial  $Q = \prod_{i \in I} (X - \beta^i)$ , where  $\beta$  is an  $n$ th primitive root of unity. To estimate the distance of such a code, we can use the following result.

**6.2.10. Theorem.** *Let  $\mathcal{C}$  be a linear cyclic code of length  $n$  over  $\mathbf{F}_q$  associated to  $I \subset (\mathbf{Z}/n\mathbf{Z})$ . If there exist  $i$  and  $s$  such that  $\{i + 1, i + 2, \dots, i + s\} \subset I$ , then  $d(\mathcal{C}) \geq s + 1$ .*

*Proof.* Let  $\beta$  be an  $n$ th primitive root in an extension of  $\mathbf{F}_q$  and let  $Q$  be a polynomial modulo  $X^n - 1$  which belongs to  $\mathcal{C}$ . We know that  $Q(\beta^{i+j}) = 0$  for  $j = 1, \dots, s$ . Assume that the weight  $w$  of  $Q$  (viewed as an element of  $\mathbf{F}_q^n$ ) is  $\leq s$ , which means that  $Q = a_1 X^{i_1} + \dots + a_w X^{i_w}$  with  $0 \leq i_1 <$

$i_2 < \cdots < i_w < n$ . We need to show that  $Q$  is in fact zero. Now, we have the equations  $a_1\beta^{i_1(i+j)} + \cdots + a_w\beta^{i_w(i+j)} = 0$  for  $j = 1, \dots, s$ . Let  $a'_1 := a_1\beta^{i_1 i}, \dots, a'_w := a_w\beta^{i_w i}$ . The equations can be rewritten as

$$\beta^{i_1 j} a'_1 + \cdots + \beta^{i_w j} a'_w = 0, \quad \text{for } j = 1, \dots, s.$$

The matrix of the  $\beta^{i_r j}$  can be extracted from a Vandermonde matrix with  $\beta^{i_r} \neq \beta^{i_{r'}}$ , because  $\beta$  has order  $n$ , and its rank therefore equals  $w = \min\{w, s\}$ . This means that  $a'_1 = \cdots = a'_w = 0$ , and hence  $a_1 = \cdots = a_w = 0$ .  $\square$

**6.2.11. Remark.** The bound given in the theorem is generally not optimal. We can see this below in the example of Golay codes.

**6.2.12. Examples.** (Linear cyclic codes.)

We will now describe in detail some examples gotten from choosing  $q, n$  and a subset  $I \subset \mathbf{Z}/n\mathbf{Z}$  which is stable under multiplication by  $q$ . To be rigorous, we should clarify that the code that we construct also depends on the  $n$ th primitive root  $\beta$  that we choose. However, it is not difficult to see that the various codes gotten from the choices of  $\beta$  are all isomorphic. We will therefore omit  $\beta$ .

**Hamming codes.** One first interesting choice of parameters is  $n = (q^r - 1)/(q - 1)$ , and we can easily check that the order of  $q \bmod n$  is  $r$ . We set  $I := \{1, q, q^2, \dots, q^{r-1}\}$ , which defines a code  $\mathcal{C}$  of dimension  $n - r$  (once  $\beta$ , a primitive  $n$ th root of unity, is chosen). We will now directly verify that  $d(\mathcal{C}) \geq 3$ . A polynomial of weight 2 can be written  $f = aX^i + bX^j$  with say  $0 \leq i < j \leq n - 1$ , and the condition that it is killed by  $\beta^{q^\ell}$  for  $0 \leq \ell \leq r - 1$  is therefore written as  $a + b\beta^{(j-i)q^\ell} = 0$ . Since  $\beta$  is of order  $n$ , we see that this is impossible except when  $a = b = 0$ . Thus the code  $\mathcal{C}$  is 1-correcting, and since  $\text{card } B(x, 1) = 1 + n(q - 1) = q^r$ , we see that  $\mathcal{C}$  is perfect 1-correcting, and thus  $d(\mathcal{C}) = 3$  or 4 (we show below that the distance is 3 and that the code is therefore MDS if and only if  $r = 2$ ). Binary Hamming codes are obtained by taking  $q = 2$  and by choosing  $I := \{1, 2, 4, \dots, 2^{r-1}\}$  and hence  $k = n - r = 2^r - r - 1$ . Since  $\{1, 2\} \subset I$ , we see that  $d(\mathcal{C}) \geq 3$ . For  $r = 3, q = 2, n = 7$ , we get the code studied in the first example (2-6.1.1).

In order to see that the distance of a Hamming code is equal to  $d(\mathcal{C}) = 3$ , we write a parity-check matrix  $A$  for the code (a matrix with  $r$  rows and  $n$  columns). The columns  $e_1, \dots, e_n$  of  $A$  are vectors in  $(\mathbf{F}_q)^r$ , and we have just shown that any pair of them is linearly independent. Now, there are  $n = (q^r - 1)/(q - 1)$  of them, and they therefore represent exactly one vector from each line in  $(\mathbf{F}_q)^r$ . Since two of the vectors  $e_i$  are never dependent,

but of course there exists triples of linearly dependent vectors, we see that  $d(\mathcal{C}) = 3$ .

**Reed-Solomon codes.** These codes correspond to the choice  $n = q - 1$ , most often with  $q = 2^f$ . Let  $\alpha$  be a generator of  $\mathbf{F}_q^*$ . Once we have chosen  $k$ , we set

$$g(X) := \prod_{i=1}^{q-1-k} (X - \alpha^i).$$

It follows of course that  $k = \dim \mathcal{C}$  and, since  $I = \{1, 2, 3, \dots, q-1-k\}$ , we have  $d(\mathcal{C}) \geq q-k$ . But we know that for every linear code,  $d(\mathcal{C}) \leq n+1-k$ , hence  $d(\mathcal{C}) = q-k$ , and the code constructed in this way is therefore MDS. Now suppose that  $q = 2^f$ . We can consider  $\mathcal{C}$  as a *binary* code  $\mathcal{C}'$ , with the parameters  $n' = (2^f - 1)f$ ,  $k' = kf$  and distance  $d(\mathcal{C}') \geq 2^f - k$ . One special feature of this code is that it can correct large numbers of errors: if  $t$  satisfies  $2t + 1 \leq d(\mathcal{C}) = q - k$ , the code can correct  $t$  elements of  $\mathbf{F}_{2^f}$ , hence  $tf$  binary errors if these errors are distributed in bunches! This feature explains why this type of code is used in the technology of compact discs.

**Ternary Golay code.** We know that  $3^5 - 1 = 11 \cdot 23$ . We choose  $q = 3$ ,  $n = 11$  and the subset of  $(\mathbf{Z}/11\mathbf{Z})^*$  generated by 3, in other words  $I := \{1, 3, 4, 5, 9\}$ ; this code, denoted by  $\mathcal{G}_{11}$ , is therefore of dimension 6. We point out (but do not use) that  $I = \mathbf{F}_{11}^{*2}$ . By Theorem 2-6.2.10 on the distance of a cyclic code, we see that  $d(\mathcal{G}_{11}) \geq 4$  and, by considering the factorization of  $\Phi_{11}$  in  $\mathbf{F}_3[X]$  (cf. Examples 2-6.2.9), we see that  $\mathcal{G}_{11}$  contains a polynomial of weight 5, hence  $d(\mathcal{G}_{11}) \leq 5$ . An extensive calculation (which is postponed to Exercise 2-7.22 below) allows us to establish that actually  $d(\mathcal{G}_{11}) = 5$ . Thus  $\mathcal{G}_{11}$  is 2-correcting, and since  $\text{card } B(x, 2) = 1 + 2\binom{11}{1} + 2^2\binom{11}{2} = 3^5$ , it is clear that the code  $\mathcal{G}_{11}$  is perfect 2-correcting (but notice that it is not MDS).

**Binary Golay code.** We know that  $2^{11} - 1 = 23 \cdot 89$  (it is actually the smallest number of the form  $2^p - 1$  which is not prime). We therefore choose  $q = 2$ ,  $n = 23$  and  $I$  as the subset of  $(\mathbf{Z}/23\mathbf{Z})^*$  generated by 2, in other words  $I := \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$ , and we denote by  $\mathcal{G}_{23}$  the associated code. Observe also that  $I = \mathbf{F}_{23}^{*2}$ . By Theorem 2-6.2.10 on the distance of a cyclic code, we see that  $d(\mathcal{G}_{23}) \geq 5$  and, by considering the factorization of  $\Phi_{23}$  in  $\mathbf{F}_2[X]$  (cf. Examples 2-6.2.9), we see that  $\mathcal{G}_{23}$  contains a polynomial of weight 7, hence  $d(\mathcal{G}_{23}) \leq 7$ . An extensive calculation (which is postponed to Exercise 2-7.22, suggested below) allows us to determine that actually  $d(\mathcal{G}_{23}) = 7$ . Thus  $\mathcal{G}_{23}$  is 3-correcting, and since  $\text{card } B(x, 3) = 1 + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 2^{11}$ , it follows that the code  $\mathcal{G}_{23}$  is perfect 3-correcting (but notice that it is not MDS).

**6.2.13. Remark.** We can show that if we exclude trivial codes (i.e., of dimension 1,  $n - 1$  or  $n$ ), the only perfect  $t$ -correcting codes are those that we have already constructed: the Hamming 1-correcting codes and the two Golay binary and ternary codes [73].

## 7. Exercises

**7.1. Exercise.** (Newton's method) Recall that Newton's iterative method (for approximating the zeros of a function) is applicable to differentiable functions. Let  $f$  be a function with a unique zero at  $\alpha$ ; the iteration is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

The rate of convergence of this approximation is quadratic, i.e.,  $|x_{n+1} - \alpha| \leq C|x_n - \alpha|^2$ . Clarify and prove this assertion for the function  $f(x) := x^m - a$ , and deduce a fast calculation algorithm for approximating  $\sqrt[m]{a}$  from this.

**7.2. Exercise.** 1) Give a fast algorithm which checks if a given integer  $N$  is a power  $a^m$ , where  $m \geq 2$ .

2) If we now want to test whether  $N = p^m$  where  $p$  is prime and  $m \geq 2$ , we take  $a \in [2, N - 1]$  and we test if  $\gcd(a, N) = 1$ . If that is the case, we compute  $d = \gcd(a^{N-1} - 1, N)$ . Prove in this case that  $p$  divides  $d$  and that, with a high probability,  $d \neq N$  and also that  $d = p$ . Deduce from this an algorithm to check whether  $N = p^m$ .

**7.3. Exercise.** (Multiplication algorithm—see [42]) Suppose that the integers  $m$  and  $n$  are written in at most  $2t$  binary digits,  $n = n_1 2^t + n_0$  and  $m = m_1 2^t + m_0$ . Observe that

$$mn = m_1 n_1 (2^{2t} - 2^t) + 2^t (m_1 + m_0)(n_1 + n_0) + m_0 n_0 (1 - 2^t)$$

and can therefore be calculated with three multiplications of numbers of size  $t$  and some additions and shifts (multiplication by 2 consists of one shift of digits). Deduce from this an algorithm, where the cost  $T(r)$  of the multiplication of two numbers with  $r$  digits satisfies

$$T(2r) \leq 3T(r) + cr,$$

for some appropriate constant  $c$ . Deduce from this that  $T(r) = O(r^\alpha)$ , where  $\alpha > \log 3 / \log 2$ . (Notice that, asymptotically, this algorithm is better than the usual algorithm, whose complexity is  $O(r^2)$ .)



**7.4. Exercise.** (Multiplication by fast Fourier transform) *In this exercise, we will give a theoretical presentation of the finite Fourier transform, which will allow us to multiply very large numbers faster than the usual algorithm. The hints are fairly brief, so you could also use a specialized reference, [42] Sect. 4.3.3., to help you finish this exercise.*

Let  $N \geq 2$  be an integer and let  $A$  be a ring. We identify the set  $E$  of functions from  $\mathbf{Z}/N\mathbf{Z}$  to  $A$  with the set of polynomials with coefficients in  $A$  of degree  $< N$ , in other words, to polynomials associated to the ring  $A[X]/(X^N - 1)$ . If  $a = (a_i)_{0 \leq i \leq N-1}$  is a sequence indexed by  $\mathbf{Z}/N\mathbf{Z}$ , we denote by  $P_a$  the corresponding polynomial. We define a “convolution” by  $(a * b)_i = \sum_{j+h=i} a_j b_h$ , and we can easily check that  $P_{a*b} = P_a P_b$ .

If  $\zeta$  is an  $N$ th primitive root of unity in  $A$ , we define the “Fourier transform”,  $\mathcal{F} : E \rightarrow E$  and its conjugate  $\bar{\mathcal{F}} : E \rightarrow E$  by the formulas

$$(\mathcal{F}a)_j = \sum_{i \in \mathbf{Z}/N\mathbf{Z}} \zeta^{ij} a_i = P_a(\zeta^j) \quad \text{and} \quad (\bar{\mathcal{F}}a)_j = \sum_{i \in \mathbf{Z}/N\mathbf{Z}} \zeta^{-ij} a_i = P_a(\zeta^{-j}).$$

1) Prove that the following formulas hold:  $\mathcal{F}(a*b) = \mathcal{F}(a) \cdot \mathcal{F}(b)$ ,  $\mathcal{F}(\bar{\mathcal{F}}a) = Na$  and  $\bar{\mathcal{F}}(\mathcal{F}a) = Na$ .

2) Whenever  $N = 2N'$ , we set  $\zeta' := \zeta^2$  and  $E' := A[X]/(X^{N'} - 1)$ , and we define  $\mathcal{F}', \bar{\mathcal{F}}' : E' \rightarrow E'$  with the help of  $\zeta'$ . For  $a \in E$ , we define  $a^0, a^1 \in E'$  by setting  $a_i^0 = a_{2i}$  and  $a_i^1 = a_{2i+1}$ . Check that, for  $0 \leq j \leq N' - 1$ , the following formulas hold:

$$(\mathcal{F}a)_j = (\mathcal{F}'a^0)_j + \zeta^j (\mathcal{F}'a^1)_j \quad \text{and} \quad (\mathcal{F}a)_{N'+j} = (\mathcal{F}'a^0)_j - \zeta^j (\mathcal{F}'a^1)_j.$$

3) Now suppose that  $N = 2^r$ . Use the previous arguments to derive a recursive procedure for calculating a Fourier transform. If we denote by  $M(r)$  the number of multiplications and  $A(r)$  the number of additions necessary to carry out this procedure, show that  $A(r) + M(r) = O(r2^r) = O(N \log N)$ .

4) By using the first formula (convolution transformation and ordinary product) and the preceding results, derive a multiplication algorithm for polynomials with coefficients in  $A$ .

5) The choice of a numeral basis  $b$  lets us write integers in the form  $P_a(b) = a_0 + a_1b + \cdots + a_db^d$ . Using the polynomial multiplication algorithm, derive an algorithm for multiplying integers.

**7.5. Exercise.** A Fibonacci sequence of integers is defined by  $u_0 = a$ ,  $u_1 = b$  and  $u_n = u_{n-1} + u_{n-2}$  for  $n \geq 2$ , where  $1 \leq a \leq b$  are integers (the classical Fibonacci sequence corresponds to  $a = b = 1$ ).

1) Prove that  $\log |u_n| \sim n \log \left( \frac{1 + \sqrt{5}}{2} \right)$ .

2) Prove that  $\gcd(u_{n+1}, u_n) = \gcd(b, a)$  and that the Euclidean algorithm gives this result in  $n$  steps. Deduce from this that the complexity estimation given at the beginning of this chapter is generally optimal.

**7.6. Exercise.** Prove that following algorithm allows us to calculate the gcd of two integers, and estimate its complexity. If  $n$  and  $m$  are even, factor out 2; if  $n$  is even and  $m$  is odd (or conversely), replace  $n$  by  $n/2$ ; if  $m$  and  $n$  are odd, replace  $n$  by  $(n - m)/2$ .

**7.7. Exercise.** Let  $M \in \mathbf{Z}$  and let  $N$  be an odd positive integer. Prove that the Euclidean algorithm, together with the quadratic reciprocity law, gives a fast algorithm (and estimate its complexity) for calculating the Jacobi symbol  $\left(\frac{M}{N}\right)$ .

**7.8. Exercise.** Let  $M := 85$ ; we define the sets  $G_0 := (\mathbf{Z}/M\mathbf{Z})^*$ ,  $G_1 := \{a \in G_0 \mid a^{M-1} = 1\}$ ,  $G_2 := \{a \in G_0 \mid a^{(M-1)/2} = \pm 1\}$ ,  $G_3 := \{a \in G_0 \mid a^{(M-1)/2} = \left(\frac{a}{M}\right)_J\}$  and finally  $S := \{a \in G_0 \mid a^{21} = 1 \text{ or } a^{21} = -1 \text{ or } a^{42} = -1\}$ .

3.a) Prove that if  $a \in S$ , then  $-a \in S$ , and use this to deduce that the cardinality of  $S$  is even.

3.b) Calculate the cardinality of  $G_0, G_1, G_2$  and  $S$ .

3.c) Use this to find the cardinality of  $G_3$ .

3.d) Is the set  $S$  a subgroup of  $G_0$ ?

**7.9. Exercise.** For  $n \geq 2$ , we denote by  $\Phi_n$  the  $n$ th cyclotomic polynomial.

1) Recall how to decompose  $\Phi_n$  in  $\mathbf{F}_p[X]$ .

2) Let  $a \in \mathbf{Z}$  and let  $p$  be a prime number which does not divide  $n$  but which divides  $\Phi_n(a)$ . Prove that  $p \equiv 1 \pmod{n}$  (you could start by observing that the class of  $a$  modulo  $p$  is a root of  $\Phi_n$ ).

3) Prove that  $\Phi_n(0) = 1$  and deduce from this that for all  $m \geq 2$ ,  $\Phi_n(m)$  is relatively prime to  $m$ . Also prove that there are only finitely many  $a \in \mathbf{Z}$  such that  $\Phi_n(a) = \pm 1$ .

4) Deduce from this (without using Dirichlet's theorem on arithmetic progressions) that there exist infinitely many prime numbers,  $p$  such that  $p \equiv 1 \pmod{n}$  (resp. infinitely many prime numbers  $p$  such that  $p \not\equiv 1 \pmod{n}$ ).

**7.10. Exercise.** Let  $G$  be a finite abelian group.

1) Prove that there exists an integer  $N$  such that  $G$  is isomorphic to a subgroup (resp. a quotient) of  $(\mathbf{Z}/N\mathbf{Z})^*$ .

Hint.— We can reduce to the case where  $G = \mathbf{Z}/n_1\mathbf{Z} \times \cdots \times \mathbf{Z}/n_s\mathbf{Z}$ . By using the result proven in the previous exercise, we can choose prime numbers  $p_i \equiv 1 \pmod{n_i}$ , and show that  $N := p_1 \cdots p_s$  works.

2) (This question requires some knowledge of Galois theory, see for example Appendix C, in particular Examples C-1.1.) Prove that there exists a finite Galois extension,  $K/\mathbf{Q}$ , such that  $\text{Gal}(K/\mathbf{Q}) \cong G$ .

**7.11. Exercise.** Let  $P = X^4 + 1$ . We will study its factorization over various fields.

1) Prove that  $P$  is irreducible in  $\mathbf{Q}[X]$  and calculate its factorization over the fields  $\mathbf{Q}(i)$ ,  $\mathbf{Q}(\sqrt{2})$  and  $\mathbf{Q}(i\sqrt{2})$ .

2) Show that for every prime number  $p$ ,  $P$  is not irreducible over  $\mathbf{F}_p$ .

Hint.— Construct a factorization by using the fact that  $-1$ ,  $2$  or  $-2$  is a square. Variation: observe that  $P = \Phi_8$  and invoke Theorem 2-6.2.8.

**7.12. Exercise.** 1) Prove that the following relations hold (you could compare the degrees and the roots of both sides):

$$\Phi_n(X^p) = \begin{cases} \Phi_{np}(X) & \text{if } p \text{ divides } n, \\ \Phi_{np}(X)\Phi_n(X) & \text{if } p \text{ does not divide } n. \end{cases}$$

2) Prove that  $\Phi_{p^r} = X^{p^{r-1}(p-1)} + X^{p^{r-1}(p-2)} + \cdots + X^{p^{r-1}} + 1$  (for  $r \geq 1$ ).

**7.13. Exercise.** For  $n \geq 3$ , we denote by  $\Phi_n^+(X)$  the monic polynomial with the property that  $(\Phi_n^+(X))^2 = \prod_{\zeta \in \mu_n^*} (X - \zeta - \zeta^{-1})$ .

1) Compute  $\Phi_3^+$ ,  $\Phi_5^+$  and  $\Phi_7^+$ .

2) Prove that  $\deg \Phi_n^+ = \phi(n)/2$  and  $\Phi_n(X) = X^{\phi(n)/2} \Phi_n^+(X + X^{-1})$ . Deduce from this that  $\Phi_p^+(2) = \Phi_p(1) = p$ .

3) Prove that  $\Phi_n^+$  is in  $\mathbf{Z}[X]$  and is irreducible (in particular, it is the minimal polynomial of  $2\cos(2\pi/n)$ ).

**7.14. Exercise.** Let  $P = \prod_{i=1}^r (X - \alpha_i)$  and  $Q = \prod_{j=1}^s (X - \beta_j)$  be two polynomials in  $K[X]$ . We define their resultant by the formula

$$\text{res}(P, Q) := \prod_{i=1}^r Q(\alpha_i) = \prod_{i=1}^r \prod_{j=1}^s (\alpha_i - \beta_j).$$

We refer you to a classical algebra text (cf. for example [43]) to see how

$\text{res}(P, Q)$  can be expressed as a determinant in the coefficients of  $P$  and  $Q$ , which shows in particular that  $\text{res}(P, Q) \in K$  and, more generally, that if  $P, Q \in A[X]$ , then  $\text{res}(P, Q) \in A$ .

1) Prove that  $\text{res}(Q, P) = (-1)^{rs} \text{res}(P, Q)$ .

2) We will assume from now on that  $P, Q \in \mathbf{Z}[X]$ , and we choose  $q$  to be an odd prime. We denote by  $\tilde{P}$  (resp.  $\tilde{Q}$ ) the reduction modulo  $q$  of  $P$  (resp. of  $Q$ ). Prove that the class of  $\text{res}(P, Q)$  modulo  $q$  is equal to  $\text{res}(\tilde{P}, \tilde{Q})$ .

3) Prove that  $\tilde{\Phi}_q^+ = (X - 2)^{(q-1)/2}$  in  $\mathbf{F}_q[X]$  ( $\Phi_n^+$  is defined in Exercise 2-7.13).

4) Use the previous questions and question 2) of Exercise 2-7.13 to show that if  $p$  and  $q$  are distinct odd primes, then

$$\text{res}(\tilde{\Phi}_q^+, \tilde{\Phi}_p^+) \equiv p^{(q-1)/2} \equiv \left(\frac{p}{q}\right) \pmod{q}.$$

5) Prove that  $\text{res}(\Phi_q^+, \Phi_p^+) = \prod_{\eta \in \mu_q^*} \eta^{-(p-1)/2} \Phi_p(\eta)$  and deduce from this that  $\text{res}(\Phi_q^+, \Phi_p^+) \in \{+1, -1\}$ .

6) Prove that the following formula holds,

$$\text{res}(\Phi_q^+, \Phi_p^+) = \left(\frac{p}{q}\right),$$

and use this to give a proof of the quadratic reciprocity law.

**7.15. Exercise.** Let  $N$  be an odd integer.

1) If its factorization can be written as  $N = p_1^{m_1} \cdots p_k^{m_k}$ , where  $p_i - 1 = 2^{s_i} L_i$  and  $L_i$  are odd, prove that

$$\frac{\text{card}\{a \in (\mathbf{Z}/N\mathbf{Z})^* \mid \text{ord}(a \bmod N) \text{ is odd}\}}{\text{card}\{a \in (\mathbf{Z}/N\mathbf{Z})^*\}} = 2^{-s_1 - \cdots - s_k}.$$

2) Deduce from this that if we had a fast algorithm,  $\mathcal{P}$ , which calculates the period (the order of  $a \bmod N$ ), then we have a fast probabilistic factorization algorithm.

Hint.— Randomly choose  $a$ , test to see whether  $\text{gcd}(a, N) = 1$ , then whether the period  $\mathcal{P}(a)$  is even; in this case compute  $\text{gcd}(a^{\mathcal{P}(a)/2} \pm 1, N)$ .

**7.16. Exercise.** Prove that  $2^m + 1$  can only be prime if  $m = 2^n$ . Set  $F_n := 2^{2^n} + 1$  (known as a Fermat number). Prove that  $F_n$  is prime if and only if  $F_n$  divides  $3^{\frac{F_n-1}{2}} + 1$ . Check that  $F_0, F_1, F_2, F_3$  and  $F_4$  are prime, but not  $F_5$  (which is divisible by 641).

**7.17. Exercise.** (Lucas test and Mersenne numbers) *Start by proving that  $M_n := 2^n - 1$  can only be prime if  $n$  is itself prime. Check that  $M_2, M_3, M_5, M_7$  are prime, but that  $M_{11}$  is not prime. The numbers  $M_p = 2^p - 1$  are called Mersenne numbers. In this exercise, we ask you to prove the Lucas primality test for these numbers.*

a) We define a sequence with values in a ring  $A$  by  $V_0 = 2$ ,  $V_1 = a$  and  $V_{n+1} - aV_n + V_{n-1} = 0$ . Verify the following formulas:  $V_{2n-1} = V_n V_{n-1} - a$ ,  $V_{2n} = V_n^2 - 2$ , and also  $V_n V_m = V_{n+m} - V_{n-m}$ .

b) Let  $M$  be odd,  $a$  an integer such that  $\gcd(a^2 - 4, M) = 1$  and  $V_n$  the sequence defined above. If  $V_{M+1} \equiv 2 \pmod{M}$  and if for every prime number  $q$  which divides  $M+1$  we have  $\gcd(V_{\frac{M+1}{q}} - 2, M) = 1$ , prove that  $M$  is prime.

c) We define the following sequence by  $L_1 := 4$  and  $L_{i+1} := L_i^2 - 2$ . Prove that the Mersenne number  $M_p$  is prime if and only if  $L_{p-1} \equiv 0 \pmod{M_p}$ .

**7.18. Exercise.** (Perfect numbers) *This nice problem has been handed down to us from Euclid: we say that an integer is perfect if it is equal to the sum of its proper divisors, symbolically:*

$$n = \sum_{\substack{d|n \\ d \neq n}} d \quad \text{or} \quad 2n = \sigma(n) := \sum_{d|n} d.$$

a) Show that if  $M_p = 2^p - 1$  is a prime Mersenne number (cf. previous exercise), then  $P_p := 2^{p-1} M_p$  is a perfect number (this fact as well as the examples  $P_2 = 6$ ,  $P_3 = 28$ ,  $P_5 = 496$  were known to Euclid).

b) Prove the following result due to Euler: an even perfect number  $n$  is of the form  $P_p$ .

Hint.— Write  $n = 2^m M$  with  $M$  odd and  $m \geq 1$ ; prove that  $2n = \sigma(2^m) \sigma(M)$  and deduce from this that  $M$  must be prime, then finish the exercise.

**Remark.** Nobody knows whether there exists an odd perfect number; it is generally conjectured that there do not exist any and that the perfect numbers are in bijection with the prime Mersenne numbers.

**7.19. Exercise.** (Pocklington-Lehmer test or certificate) *Let  $N \geq 2$ . Suppose that  $N - 1$  is (partially) factored as  $N - 1 = p_1^{e_1} \cdots p_k^{e_k} M$ , with  $M < \sqrt{N}$ , and moreover that for each  $p_i$ , we have an  $a_i$  such that*

$$\left\{ \begin{array}{l} a_i^{N-1} \equiv 1 \pmod{N}, \\ \gcd\left(a_i^{\frac{N-1}{p_i}} - 1, N\right) = 1. \end{array} \right.$$

Use this to show that if  $q$  divides  $N$ , then  $q \equiv 1 \pmod{p_i^{e_i}}$ , and also that  $N$  is prime.

**7.20. Exercise.** Let  $\beta$  be a 17th primitive root of unity in an extension of  $\mathbf{F}_2$ . We let  $I := \mathbf{F}_{17}^{*2}$  and set

$$f(X) = \prod_{i \in I} (X - \beta^i).$$

Prove that the polynomial  $f(X)$  defines a cyclic code  $\mathcal{C}$  of length 17, and calculate its dimension and bounds on its distance  $d(\mathcal{C})$ , for example  $3 \leq d(\mathcal{C}) \leq 6$ . Then give the exact value of  $d(\mathcal{C})$ .

**7.21. Exercise.** 1.a) Describe the degrees of the decomposition into irreducible factors of  $X^{85} - 1$  in  $\mathbf{Q}[X]$ .

1.b) Give the number of irreducible factors, as well as their degrees, of the decomposition of  $X^{85} - 1$  in  $\mathbf{F}_2[X]$ .

1.c) Explain how to construct a binary cyclic code of length 85 and dimension 64. It is possible to construct such a code with dimension 63?

**7.22. Exercise.** (Where we show that  $d(\mathcal{G}_{11}) = 5$  and  $d(\mathcal{G}_{23}) = 7$  and use the notion of a self-dual code.)

A) Let  $\mathcal{C}$  be a cyclic code of length  $n$  generated by the polynomial  $g = g(X)$  of degree  $d$ . Let  $\mathcal{C}'$  be its even subcode  $\mathcal{C}^*$  its dual code.

1) Prove that  $\mathcal{C}' = \mathcal{C}$  if and only if  $g(1) = 0$ . If  $g(1) \neq 0$ , check that  $\mathcal{C}'$  is cyclic and generated by the polynomial  $(X - 1)g(X)$ .

2) Prove that  $\mathcal{C}^*$  is cyclic and generated by the polynomial  $h^*(X) = X^{n-d}h(1/X)$  where  $g(X)h(X) = X^n - 1$ .

Hint.— You can show that if  $\deg(f) \leq n - d - 1$  and  $\deg(e) \leq d - 1$ , then  $\langle fg, eh^* \rangle$  is equal to the coefficient of  $X^{n-1}$  in the product  $f(X)g(X)e^*(X)h(X) = f(X)e^*(X)(X^n - 1)$ , and is therefore zero.

B) Suppose that  $\mathcal{C} \subset \mathcal{C}^*$  (i.e., for all  $x, y \in \mathcal{C}$ , we have  $\langle x, y \rangle = 0$ ).

1) If  $q = 2$ , prove that for all  $x, y \in \mathcal{C}$ , we have  $w(x + y) \equiv w(x) + w(y) \pmod{4}$ .

2) If  $q = 3$ , prove that for all  $x, y \in \mathcal{C}$ , we have  $w(x + y) \equiv w(x) + w(y) \pmod{3}$ .

C) We introduce the subcode  $\mathcal{D}$  of  $\mathcal{G}_{11}$ , composed of vectors whose sum of the coordinates equals zero (the “even” subcode).

1) Prove that if  $g(X)$  is the generating polynomial of  $\mathcal{G}_{11}$ , the code  $\mathcal{D}$  is cyclic and its generator is  $(X - 1)g(X)$ .

2) Prove that  $\mathcal{D} \subset \mathcal{D}^*$  (i.e., for every  $x, y \in \mathcal{D}$  we have  $\langle x, y \rangle = 0$ ). Deduce from this that for every  $x \in \mathcal{D}$ , we have  $w(x) \equiv 0 \pmod{3}$ .

3) We denote by  $\bar{\mathcal{D}}$  and  $\bar{\mathcal{G}}_{11}$  the extended codes. Set  $e_{11} = (1, \dots, 1) \in \mathbf{F}_3^{11}$  and  $e_{12} = (1, \dots, 1) \in \mathbf{F}_3^{12}$ . Prove that  $e_{11} \in \mathcal{G}_{11}$ ,  $e_{12} \in \mathcal{G}_{11}$  and hence  $\bar{\mathcal{G}}_{11} = \bar{\mathcal{D}} \oplus \mathbf{F}_3 e_{12}$ .

4) Prove that  $\bar{\mathcal{G}}_{11}$  is self-dual. Deduce from this that for every  $x, y \in \bar{\mathcal{G}}_{11}$ , we have  $w(x+y) \equiv w(x) + w(y) \pmod{3}$ , and hence that  $d(\bar{\mathcal{G}}_{11}) \equiv 0 \pmod{3}$ .

5) Knowing that  $4 \leq d(\mathcal{G}_{11}) \leq 5$  and  $d(C) \leq d(\bar{C}) \leq d(C) + 1$ , conclude that  $d(\mathcal{G}_{11}) = 5$  and  $d(\bar{\mathcal{G}}_{11}) = 6$ .

D) Let  $p$  be an odd prime such that  $\left(\frac{2}{p}\right) = 1$ ,  $S := \mathbf{F}_p^{*2}$  and  $\mathcal{C}$  a binary code of length  $p$  which corresponds to the set  $S$  (which, by hypothesis, is stable under multiplication by 2). We denote by  $\bar{\mathcal{C}}$  the extended code of length  $p+1$ .

1) If  $g = g(X)$  is a generator of  $\mathcal{C}$  and if  $g^*(X) = X^{(p-1)/2}g(1/X)$  is its reciprocal polynomial, show that  $g(X) = g^*(X)$  if  $p \equiv 1 \pmod{8}$ , and that  $\Phi_p(X) = g(X)g^*(X)$  if  $p \equiv -1 \pmod{8}$ .

2) We suppose from now on that  $p \equiv -1 \pmod{8}$ . Prove that  $\bar{\mathcal{C}}$  is self-dual (i.e.,  $\bar{\mathcal{C}} = \bar{\mathcal{C}}^*$ , or for all  $\bar{x}, \bar{y} \in \bar{\mathcal{C}}$ , we have  $\langle \bar{x}, \bar{y} \rangle = 0$ ).

3) Let  $x = \sum_{i \in I} X^i$  and  $y = \sum_{i \in J} X^i$ . Show that  $\langle x, y \rangle = |I \cap J| \pmod{2}$  and that  $w(x+y) = |I| + |J| - 2|I \cap J|$ . Conclude from this that if  $\langle x, y \rangle = 0$ , then  $w(x+y) \equiv w(x) + w(y) \pmod{4}$ .

4) Use the previous question to show that if  $\mathcal{D}$  is a self-dual code generated by the elements whose weight is a multiple of 4, then every element of  $\mathcal{D}$  has weight which is a multiple of 4, and in particular,  $d(\mathcal{D}) \equiv 0 \pmod{4}$ .

5) Apply the preceding questions to the case  $p = 23$ . Observe that if  $g$  is the generator of  $\mathcal{C} = \mathcal{G}_{23}$ , we have  $w(g) = 7$ , so  $w(\bar{g}) = 8$ . Conclude from this that  $d(\bar{C}) \equiv 0 \pmod{4}$ . Knowing that  $5 \leq d(\mathcal{G}_{23}) \leq 7$  and  $d(C) \leq d(\bar{C}) \leq d(C) + 1$ , deduce that  $d(\mathcal{G}_{23}) = 7$  and  $d(\bar{\mathcal{G}}_{23}) = 8$ .

Arithmetics

Hindry, M.

2011, XVIII, 322 p. 5 illus., Softcover

ISBN: 978-1-4471-2130-5