CHAPTER 1

Modular Arithmetic, Groups, Finite Fields and Probability

Chapter Goals

- To understand modular arithmetic.
- To become acquainted with groups and finite fields.
- To learn about basic techniques such as Euclid's algorithm, the Chinese Remainder Theorem and Legendre symbols.
- To recap basic ideas from probability theory.

1.1. Modular Arithmetic

Much of this book will be spent looking at the applications of modular arithmetic, since it is fundamental to modern cryptography and public key cryptosystems in particular. Hence, in this chapter we introduce the basic concepts and techniques we shall require.

The idea of modular arithmetic is essentially very simple and is identical to the "clock arithmetic" you learn in school. For example, converting between the 24-hour and the 12-hour clock systems is easy. One takes the value in the 24-hour clock system and reduces the hour by 12. For example 13:00 in the 24-hour clock system is one o'clock in the 12-hour clock system, since 13 modulo 12 is equal to one.

More formally, we fix a positive integer N which we call the *modulus*. For two integers a and b we write $a = b \pmod N$ if N divides $b - a$, and we say that a and b are *congruent* modulo N. Often we are lazy and just write $a = b$, if it is clear we are working modulo N.

We can also consider `(mod N)` as a postfix operator on an integer which returns the smallest non-negative value equal to the argument modulo N. For example

$$\begin{aligned} 18 \pmod 7 &= 4, \\ -18 \pmod 7 &= 3. \end{aligned}$$

The modulo operator is like the C operator %, except that in this book we usually take representatives which are non-negative. For example in C or Java we have,

```
(-3)%2 = -1
```

whilst we shall assume that $(-3) \pmod 2 = 1$.

For convenience we define the set

$$\mathbb{Z}/N\mathbb{Z} = \{0, \ldots, N-1\}$$

as the set of remainders modulo N. This is the set of values produced by the postfix operator `(mod N)`. Note, some authors use the alternative notation of $\mathbb{Z}_N$ for the set $\mathbb{Z}/N\mathbb{Z}$, however, in this book we shall stick to $\mathbb{Z}/N\mathbb{Z}$. For any set S we let $\#S$ denote the number of elements in the set S, thus $\#(\mathbb{Z}/N\mathbb{Z}) = N$.

N.P. Smart, *Cryptography Made Simple*, Information Security and Cryptography, DOI 10.1007/978-3-319-21936-3_1

The set $\mathbb{Z}/N\mathbb{Z}$ has two basic operations on it, namely addition and multiplication. These are defined in the obvious way, for example:

$$(11 + 13) \pmod{16} = 24 \pmod{16} = 8$$

since $24 = 1 \cdot 16 + 8$ and

$$(11 \cdot 13) \pmod{16} = 143 \pmod{16} = 15$$

since $143 = 8 \cdot 16 + 15$.

1.1.1. Groups: Addition and multiplication modulo N work almost the same as arithmetic over the reals or the integers. In particular we have the following properties:

(1) Addition is closed:
$$\forall a, b \in \mathbb{Z}/N\mathbb{Z} : \ a + b \in \mathbb{Z}/N\mathbb{Z}.$$
(2) Addition is associative:
$$\forall a, b, c \in \mathbb{Z}/N\mathbb{Z} : \ (a + b) + c = a + (b + c).$$
(3) 0 is an additive identity:
$$\forall a \in \mathbb{Z}/N\mathbb{Z} : \ a + 0 = 0 + a = a.$$
(4) The additive inverse always exists:
$$\forall a \in \mathbb{Z}/N\mathbb{Z} : \ a + (N - a) = (N - a) + a = 0,$$
i.e. $-a$ is an element which when combined with a produces the additive identity.
(5) Addition is commutative:
$$\forall a, b \in \mathbb{Z}/N\mathbb{Z} : \ a + b = b + a.$$
(6) Multiplication is closed:
$$\forall a, b \in \mathbb{Z}/N\mathbb{Z} : \ a \cdot b \in \mathbb{Z}/N\mathbb{Z}.$$
(7) Multiplication is associative:
$$\forall a, b, c \in \mathbb{Z}/N\mathbb{Z} : \ (a \cdot b) \cdot c = a \cdot (b \cdot c).$$
(8) 1 is a multiplicative identity:
$$\forall a \in \mathbb{Z}/N\mathbb{Z} : \ a \cdot 1 = 1 \cdot a = a.$$
(9) Multiplication and addition satisfy the distributive law:
$$\forall a, b, c \in \mathbb{Z}/N\mathbb{Z} : \ (a + b) \cdot c = a \cdot c + b \cdot c.$$
(10) Multiplication is commutative:
$$\forall a, b \in \mathbb{Z}/N\mathbb{Z} : \ a \cdot b = b \cdot a.$$

Many of the sets we will encounter have a number of these properties, so we give special names to these sets as a shorthand.

Definition 1.1 (Groups)**.** *A group is a set with an operation on its elements which*

- *Is closed,*
- *Has an identity,*
- *Is associative, and*
- *Every element has an inverse.*

A group which is commutative is often called *abelian*. Almost all groups that one meets in cryptography are abelian, since the commutative property is often what makes them cryptographically interesting. Hence, any set with properties 1, 2, 3 and 4 above is called a group, whilst a set with properties 1, 2, 3, 4 and 5 is called an abelian group. Standard examples of groups which one meets all the time in high school are:

- The integers, the reals or the complex numbers under addition. Here the identity is 0 and the inverse of x is $-x$, since $x + (-x) = 0$.
- The non-zero rational, real or complex numbers under multiplication. Here the identity is 1 and the inverse of x is denoted by x^{-1}, since $x \cdot x^{-1} = 1$.

A group is called *multiplicative* if we tend to write its group operation in the same way as one does for multiplication, i.e.

$$f = g \cdot h \text{ and } g^5 = g \cdot g \cdot g \cdot g \cdot g.$$

We use the notation $(G, \cdot)$ in this case if there is some ambiguity as to which operation on G we are considering. A group is called *additive* if we tend to write its group operation in the same way as one does for addition, i.e.

$$f = g + h \text{ and } 5 \cdot g = g + g + g + g + g.$$

In this case we use the notation $(G, +)$ if there is some ambiguity. An abelian group is called *cyclic* if there is a special element, called the *generator*, from which every other element can be obtained either by repeated application of the group operation, or by the use of the inverse operation. For example, in the integers under addition every positive integer can be obtained by repeated addition of 1 to itself, e.g. 7 can be expressed by

$$7 = 1 + 1 + 1 + 1 + 1 + 1 + 1.$$

Every negative integer can be obtained from a positive integer by application of the additive inverse operator, which sends x to $-x$. Hence, we have that 1 is a generator of the integers under addition.

If g is a generator of the cyclic group G we often write $G = \langle g \rangle$. If G is multiplicative then every element h of G can be written as

$$h = g^x,$$

whilst if G is additive then every element h of G can be written as

$$h = x \cdot g,$$

where x in both cases is some integer called the *discrete logarithm* of h to the base g.

1.1.2. Rings: As well as groups we also use the concept of a ring.

Definition 1.2 (Rings)**.** *A ring is a set with two operations, usually denoted by $+$ and $\cdot$ for addition and multiplication, which satisfies properties* 1 *to* 9 *above. We can denote a ring and its two operations by the triple $(R, \cdot, +)$. If it also happens that multiplication is commutative we say that the ring is commutative.*

This may seem complicated but it sums up the type of sets one deals with all the time, for example the infinite commutative rings of integers, real or complex numbers. In fact in cryptography things are even easier since we only need to consider finite rings, like the commutative ring of integers modulo N, $\mathbb{Z}/N\mathbb{Z}$. Thus $\mathbb{Z}/N\mathbb{Z}$ is an abelian group when we only think of addition, but it is also a ring if we want to worry about multiplication as well.

1.1.3. Euler's ϕ Function: In modular arithmetic it will be important to know when, given a and b, the equation

$$a \cdot x = b \pmod{N}$$

has a solution. For example there is exactly one solution in the set $\mathbb{Z}/143\mathbb{Z} = \{0, \ldots, 142\}$ to the equation

$$7 \cdot x = 3 \pmod{143},$$

but there are no solutions to the equation

$$11 \cdot x = 3 \pmod{143},$$

however there are 11 solutions to the equation

$$11 \cdot x = 22 \pmod{143}.$$

Luckily, it is very easy to test when such an equation has one, many or no solutions. We simply compute the greatest common divisor, or gcd, of a and N, i.e. $\gcd(a, N)$.

- If $\gcd(a, N) = 1$ then there is exactly one solution. We find the value c such that $a \cdot c = 1 \pmod{N}$ and then we compute $x \leftarrow b \cdot c \pmod{N}$.
- If $g = \gcd(a, N) \neq 1$ and $\gcd(a, N)$ divides b then there are g solutions. Here we divide the whole equation by g to produce the equation

  $$a' \cdot x' = b' \pmod{N'},$$

  where $a' = a/g$, $b' = b/g$ and $N' = N/g$. If x' is a solution to the above equation then

  $$x \leftarrow x' + i \cdot N'$$

  for $0 \leq i < g$ is a solution to the original one.
- Otherwise there are no solutions.

The case where $\gcd(a, N) = 1$ is so important we have a special name for it: we say a and N are relatively prime or coprime.

In the above description we wrote $x \leftarrow y$ to mean that we *assign* x the value y; this is to distinguish it from saying $x = y$, by which we mean x and y are equal. Clearly after assignment of y to x the values of x and y are indeed equal. But imagine we wanted to increment x by one, we would write $x \leftarrow x + 1$, the meaning of which is clear. Whereas $x = x + 1$ is possibly a statement which evaluates to false!

Another reason for this special notation for assignment is that we can extend it to algorithms, or procedures. So for example $x \leftarrow A(z)$ might mean we assign x the output of procedure A on input of z. This procedure might be randomized, and in such a case we are thereby assuming an implicit probability distribution of the output x. We might even write $x \leftarrow S$ where S is some set, by which we mean we assign x a value from the set S chosen uniformly at random. Thus our original $x \leftarrow y$ notation is just a shorthand for $x \leftarrow \{y\}$.

The number of integers in $\mathbb{Z}/N\mathbb{Z}$ which are relatively prime to N is given by the Euler ϕ function, $\phi(N)$. Given the prime factorization of N it is easy to compute the value of $\phi(N)$. If N has the prime factorization

$$N = \prod_{i=1}^{n} p_i^{e_i}$$

then

$$\phi(N) = \prod_{i=1}^{n} p_i^{e_i - 1}(p_i - 1).$$

Note, the last statement is very important for cryptography: *Given the factorization of N it is easy to compute the value of $\phi(N)$.* The most important cases for the value of $\phi(N)$ in cryptography are:

(1) If p is prime then

$$\phi(p) = p - 1.$$

(2) If p and q are both prime and $p \neq q$ then

$$\phi(p \cdot q) = (p - 1)(q - 1).$$

1.1.4. Multiplicative Inverse Modulo N: We have just seen that when we wish to solve equations of the form

$$a \cdot x = b \pmod{N}$$

we reduce the problem to the question of examining whether a has a multiplicative inverse modulo N, i.e. whether there is a number c such that

$$a \cdot c = c \cdot a = 1 \pmod{N}.$$

Such a value of c is often written a^{-1}. Clearly a^{-1} is the solution to the equation

$$a \cdot x = 1 \pmod{N}.$$

Hence, the inverse of a only exists when a and N are coprime, i.e. $\gcd(a, N) = 1$. Of particular interest is when N is a prime p, since then for all non-zero values of $a \in \mathbb{Z}/p\mathbb{Z}$ we always obtain a unique solution to

$$a \cdot x = 1 \pmod{p}.$$

Hence, if p is a prime then every non-zero element in $\mathbb{Z}/p\mathbb{Z}$ has a multiplicative inverse. A ring like $\mathbb{Z}/p\mathbb{Z}$ with this property is called a field.

Definition 1.3 (Fields)**.** *A field is a set with two operations $(G, \cdot, +)$ such that*

- *$(G, +)$ is an abelian group with identity denoted by 0,*
- *$(G \setminus \{0\}, \cdot)$ is an abelian group,*
- *$(G, \cdot, +)$ satisfies the distributive law.*

Hence, a field is a commutative ring for which every non-zero element has a multiplicative inverse. You have met fields before, for example consider the infinite fields of rational, real or complex numbers.

1.1.5. The Set $(\mathbb{Z}/N\mathbb{Z})^*$: We define the set of all invertible elements in $\mathbb{Z}/N\mathbb{Z}$ by

$$(\mathbb{Z}/N\mathbb{Z})^* = \{x \in \mathbb{Z}/N\mathbb{Z} : \gcd(x, N) = 1\}.$$

The * in A^*, for any ring A, refers to the largest subset of A which forms a group under multiplication. Hence, the set $(\mathbb{Z}/N\mathbb{Z})^*$ is a group with respect to multiplication and it has size $\phi(N)$. In the special case when N is a prime p we have

$$(\mathbb{Z}/p\mathbb{Z})^* = \{1, \ldots, p-1\}$$

since every non-zero element of $\mathbb{Z}/p\mathbb{Z}$ is coprime to p. For an arbitrary field F the set F^* is equal to the set $F \setminus \{0\}$. To ease notation, for this very important case, we define

$$\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z} = \{0, \ldots, p-1\}$$

and

$$\mathbb{F}_p^* = (\mathbb{Z}/p\mathbb{Z})^* = \{1, \ldots, p-1\}.$$

The set $\mathbb{F}_p$ is said to be a finite field of characteristic p. In the next section we shall discuss a more general type of finite field, but for now recall the important point that the integers modulo N are only a field when N is a prime. We end this section with the most important theorem in elementary group theory.

Theorem 1.4 (Lagrange's Theorem)**.** *If $(G, \cdot)$ is a group of order (size) $n = \#G$ then for all $a \in G$ we have $a^n = 1$.*

So if $x \in (\mathbb{Z}/N\mathbb{Z})^*$ then

$$x^{\phi(N)} = 1 \pmod{N}$$

since $\#(\mathbb{Z}/N\mathbb{Z})^* = \phi(N)$. This leads us to Fermat's Little Theorem, not to be confused with Fermat's Last Theorem which is something entirely different.

Theorem 1.5 (Fermat's Little Theorem)**.** *Suppose p is a prime and $a \in \mathbb{Z}$, then*

$$a^p = a \pmod{p}.$$

Fermat's Little Theorem is a special case of Lagrange's Theorem and will form the basis of one of the primality tests considered in a later chapter.

1.2. Finite Fields

The integers modulo a prime p are not the only type of finite field. In this section we shall introduce another type of finite field which is particularly important. At first reading you may wish to skip this section. We shall only be using these general forms of finite fields when discussing the AES block cipher, stream ciphers based on linear feedback shift registers and when we look at systems based on elliptic curves.

For this section we let p denote a prime number. Consider the set of polynomials in X whose coefficients are elements of $\mathbb{F}_p$. We denote this set $\mathbb{F}_p[X]$, which forms a ring with the natural definition of addition and multiplication of polynomials modulo p. Of particular interest is the case when $p = 2$, from which we draw most of our examples in this section. For example, in $\mathbb{F}_2[X]$ we have

$$\begin{aligned}(1 + X + X^2) + (X + X^3) &= 1 + X^2 + X^3,\\ (1 + X + X^2) \cdot (X + X^3) &= X + X^2 + X^4 + X^5.\end{aligned}$$

Just as with the integers modulo a number N, where the integers modulo N formed a ring, we can take a polynomial $f(X)$ and then the polynomials modulo $f(X)$ also form a ring. We denote this ring by

$$\mathbb{F}_p[X]/f(X)\mathbb{F}_p[X]$$

or more simply

$$\mathbb{F}_p[X]/(f(X)).$$

But to ease notation we will often write $\mathbb{F}_p[X]/f(X)$ for this latter ring. When $f(X) = X^4 + 1$ and $p = 2$ we have, for example,

$$(1 + X + X^2) \cdot (X + X^3) \pmod{X^4 + 1} = 1 + X^2$$

since

$$X + X^2 + X^4 + X^5 = (X + 1) \cdot (X^4 + 1) + (1 + X^2).$$

When checking the above equation you should remember we are working modulo two.

1.2.1. Inversion in General Finite Fields: Recall, when we looked at the integers modulo N we looked at the equation $a \cdot x = b \pmod{N}$. We can consider a similar question for polynomials. Given a, b and f, all of which are polynomials in $\mathbb{F}_p[X]$, does there exist a solution α to the equation $a \cdot \alpha = b \pmod{f}$? With integers the answer depended on the greatest common divisor of a and f, and we counted three possible cases. A similar three cases can occur for polynomials, with the most important one being when a and f are coprime and so have greatest common divisor equal to one.

A polynomial is called irreducible if it has no proper factors other than itself and the constant polynomials. Hence, irreducibility of polynomials is the same as primality of numbers. Just as with the integers modulo N, when N was prime we obtained a finite field, so when $f(X)$ is irreducible the ring $\mathbb{F}_p[X]/f(X)$ also forms a finite field.

1.2.2. Isomorphisms of Finite Fields: Consider the case $p = 2$ and the two different irreducible polynomials

$$f_1 = X^7 + X + 1$$

and

$$f_2 = Y^7 + Y^3 + 1.$$

Now, consider the two finite fields

$$F_1 = \mathbb{F}_2[X]/f_1(X) \text{ and } F_2 = \mathbb{F}_2[Y]/f_2(Y).$$

These both consist of the 2^7 binary polynomials of degree less than seven. Addition in these two fields is identical in that one just adds the coefficients of the polynomials modulo two. The only difference is in how multiplication is performed

$$\begin{aligned}(X^3 + 1) \cdot (X^4 + 1) \pmod{f_1(X)} &= X^4 + X^3 + X,\\ (Y^3 + 1) \cdot (Y^4 + 1) \pmod{f_2(Y)} &= Y^4.\end{aligned}$$

A natural question arises as to whether these fields are "really" different, or whether they just "look" different. In mathematical terms the question is whether the two fields are *isomorphic*. It turns out that they are isomorphic if there is a map

$$\phi : F_1 \longrightarrow F_2,$$

called a field isomorphism, which satisfies

$$\begin{aligned}\phi(\alpha + \beta) &= \phi(\alpha) + \phi(\beta),\\ \phi(\alpha \cdot \beta) &= \phi(\alpha) \cdot \phi(\beta).\end{aligned}$$

Such an isomorphism exists for every two finite fields of the same order, although we will not show it here. To describe the map above you only need to show how to express a root of $f_2(Y)$ in terms of a polynomial in the root of $f_1(X)$, with the inverse map being a polynomial which expresses a root of $f_1(X)$ in terms of a polynomial in the root of $f_2(Y)$, i.e.

$$\begin{aligned}Y = g_1(X) &= X + X^2 + X^3 + X^5,\\ X = g_2(Y) &= Y^5 + Y^4.\end{aligned}$$

Notice that $g_2(g_1(X)) \pmod{f_1(X)} = X$, that $f_2(g_1(X)) \pmod{f_1(X)} = 0$ and that $f_1(g_2(Y)) \pmod{f_2(Y)} = 0$.

One can show that all finite fields of the same characteristic and prime are isomorphic, thus we have the following.

Theorem 1.6. *There is (up to isomorphism) just one finite field of each prime power order.*

The notation we use for these fields is either $\mathbb{F}_q$ or $GF(q)$, with $q = p^d$ where d is the degree of the irreducible polynomial used to construct the field; we of course have $\mathbb{F}_p = \mathbb{F}_p[X]/X$. The notation $GF(q)$ means the Galois field of q elements, in honour of the nineteenth century French mathematician Galois. Galois had an interesting life; he accomplished his scientific work at an early age before dying in a duel.

1.2.3. Field Towers and the Frobenius Map: There are a number of technical definitions associated with finite fields which we need to cover. A subset F of a field K is called a *subfield* if F is a field with respect to the same operations for which K is a field. Each finite field K contains a copy of the integers modulo p for some prime p, i.e. $\mathbb{F}_p \subset K$. We call this prime the *characteristic* of the field, and often write this as char K. The subfield of integers modulo p of a finite field is called the prime subfield.

There is a map Φ called the pth power *Frobenius map* defined for any finite field by

$$\Phi : \begin{cases} \mathbb{F}_q \longrightarrow \mathbb{F}_q \\ \alpha \longmapsto \alpha^p \end{cases}$$

where p is the characteristic of $\mathbb{F}_q$. The Frobenius map is an isomorphism of $\mathbb{F}_q$ with itself; such an isomorphism is called an automorphism. An interesting property is that the set of elements fixed by the Frobenius map is the prime field, i.e.

$$\{\alpha \in \mathbb{F}_q : \alpha^p = \alpha\} = \mathbb{F}_p.$$

Notice that this is a kind of generalization of Fermat's Little Theorem to finite fields. For any automorphism χ of a finite field, the set of elements fixed by χ is a field, called the fixed field of χ. Hence the previous statement says that the fixed field of the Frobenius map is the prime field $\mathbb{F}_p$.

Not only does $\mathbb{F}_q$ contain a copy of $\mathbb{F}_p$ but $\mathbb{F}_{p^d}$ contains a copy of $\mathbb{F}_{p^e}$ for every value of e dividing d; see Figure 1.1 for an example. In addition $\mathbb{F}_{p^e}$ is the fixed field of the automorphism Φ^e, i.e.

$$\{\alpha \in \mathbb{F}_{p^d} : \alpha^{p^e} = \alpha\} = \mathbb{F}_{p^e}.$$

If we define $\mathbb{F}_q$ as $\mathbb{F}_p[X]/f(X)$, for some irreducible polynomial $f(X)$ with $p^{\deg f} = q$, then another

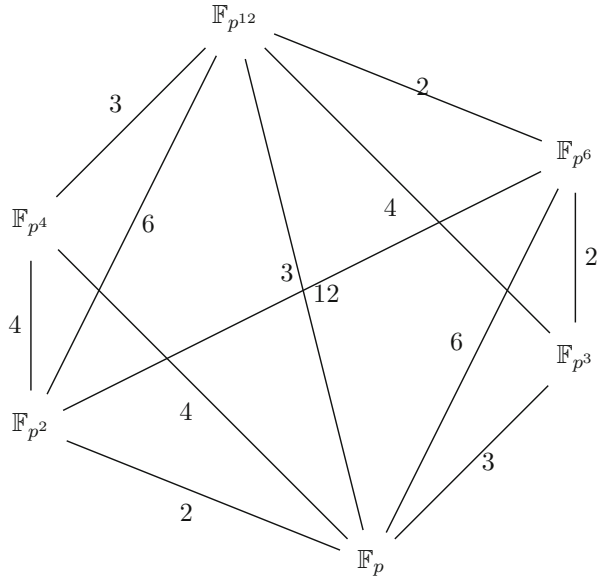

FIGURE 1.1. Example tower of finite fields. The number on each line gives the degree of the subfield within the larger field

way of thinking of $\mathbb{F}_q$ is as the set of polynomials of degree less than $\deg f$ in a root of $f(X)$. In other words let α be a "formal" root of $f(X)$, then we define

$$\mathbb{F}_q = \{ \sum_{i=0}^{\deg f - 1} a_i \cdot \alpha^i : a_i \in \mathbb{F}_p \}$$

with addition being addition of polynomials modulo p, and multiplication being polynomial multiplication modulo p, subject to the fact that $f(\alpha) = 0$. To see why this amounts to the same object

take two polynomials $a(X)$ and $b(X)$ and let $c(X) = a(X) \cdot b(X) \pmod{f(X)}$. Then there is a polynomial $q(X)$ such that

$$c(X) = a(X) \cdot b(X) + q(X) \cdot f(X),$$

which is our multiplication method given in terms of polynomials. In terms of a root α of $f(X)$ we note that we have

$$\begin{aligned} c(\alpha) &= a(\alpha) \cdot b(\alpha) + q(\alpha) \cdot f(\alpha). \\ &= a(\alpha) \cdot b(\alpha) + q(\alpha) \cdot 0, \\ &= a(\alpha) \cdot b(\alpha). \end{aligned}$$

Another interesting property is that if p is the characteristic of $\mathbb{F}_q$ then if we take any element $\alpha \in \mathbb{F}_q$ and add it to itself p times we obtain zero, e.g. in $\mathbb{F}_{49}$ we have

$$X + X + X + X + X + X + X = 7 \cdot X = 0 \pmod{7}.$$

The non-zero elements of a finite field, usually denoted $\mathbb{F}_q^*$, form a cyclic finite abelian group, called the multiplicative group of the finite field. We call a generator of $\mathbb{F}_q^*$ a primitive element in the finite field. Such primitive elements always exist, and indeed there are $\phi(q)$ of them, and so the multiplicative group is always cyclic. In other words there always exists an element $g \in \mathbb{F}_q$ such that every non-zero element α can be written as

$$\alpha = g^x$$

for some integer value of x.

Example: As an example consider the field of eight elements defined by

$$\mathbb{F}_{2^3} = \mathbb{F}_2[X]/(X^3 + X + 1).$$

In this field there are seven non-zero elements; namely

$$1, \alpha, \alpha + 1, \alpha^2, \alpha^2 + 1, \alpha^2 + \alpha, \alpha^2 + \alpha + 1$$

where α is a root of $X^3 + X + 1$. We see that α is a primitive element in $\mathbb{F}_{2^3}$ since

$$\begin{aligned} \alpha^1 &= \alpha, \\ \alpha^2 &= \alpha^2, \\ \alpha^3 &= \alpha + 1, \\ \alpha^4 &= \alpha^2 + \alpha, \\ \alpha^5 &= \alpha^2 + \alpha + 1, \\ \alpha^6 &= \alpha^2 + 1, \\ \alpha^7 &= 1. \end{aligned}$$

Notice that for a prime p this means that the integers modulo p also have a primitive element, since $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ is a finite field.

1.3. Basic Algorithms

There are several basic numerical algorithms or techniques which everyone should know since they occur in many places in this book. The ones we shall concentrate on here are

- Euclid's gcd algorithm,
- The Chinese Remainder Theorem,
- Computing Jacobi and Legendre symbols.

1.3.1. Greatest Common Divisors: In the previous sections we said that when trying to solve

$$a \cdot x = b \pmod{N}$$

in integers, or

$$a \cdot \alpha = b \pmod{f}$$

for polynomials modulo a prime, we needed to compute the greatest common divisor. This was particularly important in determining whether $a \in \mathbb{Z}/N\mathbb{Z}$ or $a \in \mathbb{F}_p[X]/f$ had a multiplicative inverse or not, i.e. $\gcd(a, N) = 1$ or $\gcd(a, f) = 1$. We did not explain how this greatest common divisor is computed, neither did we explain how the inverse is to be computed when we know it exists. We shall now address this omission by explaining one of the oldest algorithms known to man, namely the Euclidean algorithm.

If we were able to factor a and N into primes, or a and f into irreducible polynomials, then computing the greatest common divisor would be particularly easy. For example if we were given

$$\begin{aligned} a &= 230\,895\,588\,646\,864 = 2^4 \cdot 157 \cdot 4513^3, \\ b &= 33\,107\,658\,350\,407\,876 = 2^2 \cdot 157 \cdot 2269^3 \cdot 4513, \end{aligned}$$

then it is easy, from the factorization, to compute the gcd as

$$\gcd(a, b) = 2^2 \cdot 157 \cdot 4513 = 2\,834\,164.$$

However, factoring is an expensive operation for integers, so the above method is very slow for large integers. However, computing greatest common divisors is actually easy as we shall now show. Although factoring for polynomials modulo a prime is very easy, it turns out that almost all algorithms to factor polynomials require access to an algorithm to compute greatest common divisors. Hence, in both situations we need to be able to compute greatest common divisors without recourse to factoring.

1.3.2. The Euclidean Algorithm: In the following we will consider the case of integers only; the generalization to polynomials is easy since both integers and polynomials allow Euclidean division. For integers a and b, Euclidean division is the operation of finding q and r with $0 \leq r < |b|$ such that

$$a = q \cdot b + r,$$

i.e. $r \leftarrow a \pmod{b}$. For polynomials f and g, Euclidean division means finding polynomials q, r with $0 \leq \deg r < \deg g$ such that

$$f = q \cdot g + r.$$

To compute the gcd of $r_0 = a$ and $r_1 = b$ we compute $r_2, r_3, r_4, \ldots$ by $r_{i+2} = r_i \pmod{r_{i+1}}$, until $r_{m+1} = 0$, so we have:

$$\begin{aligned} r_2 &\leftarrow r_0 - q_1 \cdot r_1, \\ r_3 &\leftarrow r_1 - q_2 \cdot r_2, \\ &\vdots \quad \vdots \\ r_m &\leftarrow r_{m-2} - q_{m-1} \cdot r_{m-1}, \\ r_{m+1} &\leftarrow 0, \qquad\qquad \text{i.e. } r_m \text{ divides } r_{m-1}. \end{aligned}$$

If d divides a and b then d divides r_2, r_3, r_4 and so on. Hence

$$\gcd(a, b) = \gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m.$$

As an example of this algorithm we can show that $3 = \gcd(21, 12)$. Using the Euclidean algorithm we compute $\gcd(21, 12)$ in the steps

$$\begin{aligned}\gcd(21, 12) &= \gcd(21 \pmod{12}, 12)\\ &= \gcd(9, 12)\\ &= \gcd(12 \pmod{9}, 9)\\ &= \gcd(3, 9)\\ &= \gcd(9 \pmod{3}, 3)\\ &= \gcd(0, 3) = 3.\end{aligned}$$

Or, as an example with larger numbers,

$$\begin{aligned}\gcd(1\,426\,668\,559\,730,\ 810\,653\,094\,756) &= \gcd(810\,653\,094\,756,\ 616\,015\,464\,974),\\ &= \gcd(616\,015\,464\,974,\ 194\,637\,629\,782),\\ &= \gcd(194\,637\,629\,782,\ 32\,102\,575\,628),\\ &= \gcd(32\,102\,575\,628,\ 2\,022\,176\,014),\\ &= \gcd(2\,022\,176\,014,\ 1\,769\,935\,418),\\ &= \gcd(1\,769\,935\,418,\ 252\,240\,596),\\ &= \gcd(252\,240\,596,\ 4\,251\,246),\\ &= \gcd(4\,251\,246,\ 1\,417\,082),\\ &= \gcd(1\,417\,082,\ 0),\\ &= 1\,417\,082.\end{aligned}$$

The Euclidean algorithm essentially works because the mapping

$$(a, b) \longmapsto (a \pmod{b}, b),$$

for $a \geq b$ is a gcd-preserving mapping, i.e. the input and output of pairs of integers from the mapping have the same greatest common divisor. In computer science terms the greatest common divisor is an invariant of the mapping. In addition for inputs $a, b > 0$ the algorithm terminates since the mapping produces a sequence of decreasing non-negative integers, which must eventually end up with the smallest value being zero.

The trouble with the above method for determining a greatest common divisor is that computers find it much easier to add and multiply numbers than to take remainders or quotients. Hence, implementing a gcd algorithm with the above gcd-preserving mapping will usually be very inefficient. Fortunately, there are a number of other gcd-preserving mappings: For example the following is a gcd-preserving mapping between pairs of integers, which are not both even,

$$(a, b) \longmapsto \begin{cases}((a-b)/2, b) & \text{If } a \text{ and } b \text{ are odd.}\\ (a/2, b) & \text{If } a \text{ is even and } b \text{ is odd.}\\ (a, b/2) & \text{If } a \text{ is odd and } b \text{ is even.}\end{cases}$$

Recall that computers find it easy to divide by two, since in binary this is accomplished by a cheap bit shift operation. This latter mapping gives rise to the binary Euclidean algorithm, which is the one usually implemented on a computer. Essentially, this algorithm uses the above gcd-preserving mapping after first removing any power of two in the gcd. Algorithm 1.1 explains how this works, on input of two positive integers a and b.

Algorithm 1.1: Binary Euclidean algorithm

```
g ← 1.
/* Remove powers of two from the gcd */
while (a mod 2 = 0) and (b mod 2 = 0) do
    a ← a/2, b ← b/2, g ← 2 · g.
/* At least one of a and b is now odd */
while a ≠ 0 do
    while a mod 2 = 0 do a ← a/2.
    while b mod 2 = 0 do b ← b/2.
    /* Now both a and b are odd */
    if a ≥ b then a ← (a − b)/2.
    else b ← (b − a)/2.
return g · b
```

1.3.3. The Extended Euclidean Algorithm: Using the Euclidean algorithm we can determine when a has an inverse modulo N by testing whether

$$\gcd(a, N) = 1.$$

But we still do not know how to determine the inverse when it exists. To do this we use a variant of Euclid's gcd algorithm, called the extended Euclidean algorithm. Recall we had

$$r_{i-2} = q_{i-1} \cdot r_{i-1} + r_i$$

with $r_m = \gcd(r_0, r_1)$. Now we unwind the above and write each r_i, for $i \geq 2$, in terms of a and b. So we have the identities

$$\begin{aligned}
r_2 &= r_0 - q_1 \cdot r_1 = a - q_1 \cdot b \\
r_3 &= r_1 - q_2 \cdot r_2 = b - q_2 \cdot (a - q_1 \cdot b) = -q_2 \cdot a + (1 + q_1 \cdot q_2) \cdot b \\
&\vdots \quad \vdots \\
r_{i-2} &= s_{i-2} \cdot a + t_{i-2} \cdot b \\
r_{i-1} &= s_{i-1} \cdot a + t_{i-1} \cdot b \\
r_i &= r_{i-2} - q_{i-1} \cdot r_{i-1} \\
&= a \cdot (s_{i-2} - q_{i-1} \cdot s_{i-1}) + b \cdot (t_{i-2} - q_{i-1} \cdot t_{i-1}) \\
&\vdots \quad \vdots \\
r_m &= s_m \cdot a + t_m \cdot b.
\end{aligned}$$

The extended Euclidean algorithm takes as input a and b and outputs values r_m, s_m and t_m such that

$$r_m = \gcd(a, b) = s_m \cdot a + t_m \cdot b.$$

Hence, we can now solve our original problem of determining the inverse of a modulo N, when such an inverse exists. We first apply the extended Euclidean algorithm to a and $b = N$ so as to compute d, x, y such that

$$d = \gcd(a, N) = x \cdot a + y \cdot N.$$

This algorithm is described in Algorithm 1.2. The value d will be equal to one, as we have assumed that a and N are coprime. Given the output from this algorithm, we can solve the equation $a \cdot x = 1 \pmod{N}$, since we have $d = x \cdot a + y \cdot N = x \cdot a \pmod{N}$.

Algorithm 1.2: Extended Euclidean algorithm

$s \leftarrow 0, s' \leftarrow 1, t \leftarrow 1, t' \leftarrow 0, r \leftarrow b, r' \leftarrow a.$
while $r \neq 0$ **do**
  $q \leftarrow \lfloor r'/r \rfloor.$
  $(r', r) \leftarrow (r, r' - q \cdot r).$
  $(s', s) \leftarrow (s, s' - q \cdot s).$
  $(t', t) \leftarrow (t, t' - q \cdot t).$
$d \leftarrow r', x \leftarrow t, y \leftarrow s.$
return d, x, y.

As an example suppose we wish to compute the inverse of 7 modulo 19. We first set $r_0 = 7$ and $r_1 = 19$ and then we compute

$$\begin{aligned} r_2 &\leftarrow 5 = 19 - 2 \cdot 7 \\ r_3 &\leftarrow 2 = 7 - 5 = 7 - (19 - 2 \cdot 7) = -19 + 3 \cdot 7 \\ r_4 &\leftarrow 1 = 5 - 2 \cdot 2 = (19 - 2 \cdot 7) - 2 \cdot (-19 + 3 \cdot 7) = 3 \cdot 19 - 8 \cdot 7. \end{aligned}$$

Hence,

$$1 = -8 \cdot 7 \pmod{19}$$

and so

$$7^{-1} = -8 = 11 \pmod{19}.$$

Note, a binary version of the above algorithm also exists. We leave it to the reader to work out the details of the binary version of the extended Euclidean algorithm.

1.3.4. Chinese Remainder Theorem (CRT): The Chinese Remainder Theorem, or CRT, is also a very old piece of mathematics, which dates back at least 2 000 years. We shall use the CRT in a few places, for example to improve the performance of the decryption operation of RSA and in a number of other protocols. In a nutshell the CRT states that if we have the two equations

$$x = a \pmod{N} \text{ and } x = b \pmod{M}$$

then there is a unique solution modulo $(M \cdot N)$ if and only if $\gcd(N, M) = 1$. In addition it gives a method to easily find the solution. For example if the two equations are given by

$$\begin{aligned} x &= 4 \pmod{7}, \\ x &= 3 \pmod{5}, \end{aligned}$$

then we have

$$x = 18 \pmod{35}.$$

It is easy to check that this is a solution, since $18 \pmod 7 = 4$ and $18 \pmod 5 = 3$. But how did we produce this solution?

We shall first show how this can be done naively from first principles and then we shall give the general method. We have the equations

$$x = 4 \pmod{7} \text{ and } x = 3 \pmod{5}.$$

Hence for some u we have

$$x = 4 + 7 \cdot u \text{ and } x = 3 \pmod{5}.$$

Putting these latter two equations together, one obtains

$$4 + 7 \cdot u = 3 \pmod{5}.$$

We then rearrange the equation to find

$$2 \cdot u = 7 \cdot u = 3 - 4 = 4 \pmod 5.$$

Now since $\gcd(2, 5) = 1$ we can solve the above equation for u. First we compute $2^{-1} \pmod 5 = 3$, since $2 \cdot 3 = 6 = 1 \pmod 5$. Then we compute the value of $u = 2^{-1} \cdot 4 = 3 \cdot 4 = 2 \pmod 5$. Then substituting this value of u back into our equation for x gives the solution

$$x = 4 + 7 \cdot u = 4 + 7 \cdot 2 = 18.$$

The Chinese Remainder Theorem: Two Equations: The case of two equations is so important we now give a general formula. We assume that $\gcd(N, M) = 1$, and that we are given the equations

$$x = a \pmod M \text{ and } x = b \pmod N.$$

We first compute

$$T \leftarrow M^{-1} \pmod N$$

which is possible since we have assumed $\gcd(N, M) = 1$. We then compute

$$u \leftarrow (b - a) \cdot T \pmod N.$$

The solution modulo $M \cdot N$ is then given by

$$x \leftarrow a + u \cdot M.$$

To see this always works we verify

$$\begin{aligned} x \pmod M &= a + u \cdot M \pmod M \\ &= a, \\ x \pmod N &= a + u \cdot M \pmod N \\ &= a + (b - a) \cdot T \cdot M \pmod N \\ &= a + (b - a) \cdot M^{-1} \cdot M \pmod N \\ &= a + (b - a) \pmod N \\ &= b. \end{aligned}$$

The Chinese Remainder Theorem: The General Case: Now we turn to the general case of the CRT where we consider more than two equations at once. Let $m_1, \ldots, m_r$ be pairwise relatively prime and let $a_1, \ldots, a_r$ be given. We want to find x modulo $M = m_1 \cdot m_2 \cdots m_r$ such that

$$x = a_i \pmod{m_i} \text{ for all } i.$$

The Chinese Remainder Theorem guarantees a unique solution given by

$$x \leftarrow \sum_{i=1}^{r} a_i \cdot M_i \cdot y_i \pmod M$$

where

$$M_i \leftarrow M/m_i \text{ and } y_i \leftarrow M_i^{-1} \pmod{m_i}.$$

As an example suppose we wish to find the unique x modulo

$$M = 1001 = 7 \cdot 11 \cdot 13$$

such that

$$\begin{aligned} x &= 5 \pmod 7, \\ x &= 3 \pmod{11}, \\ x &= 10 \pmod{13}. \end{aligned}$$

We compute

$$\begin{aligned} M_1 &\leftarrow 143, \quad y_1 \leftarrow 5, \\ M_2 &\leftarrow 91, \quad y_2 \leftarrow 4, \\ M_3 &\leftarrow 77, \quad y_3 \leftarrow 12. \end{aligned}$$

Then, the solution is given by

$$\begin{aligned} x &\leftarrow \sum_{i=1}^{r} a_i \cdot M_i \cdot y_i \pmod{M} \\ &= 715 \cdot 5 + 364 \cdot 3 + 924 \cdot 10 \pmod{1001} \\ &= 894. \end{aligned}$$

1.3.5. The Legendre Symbol: Let p denote a prime, greater than two. Consider the mapping

$$\begin{aligned} \mathbb{F}_p &\longrightarrow \mathbb{F}_p \\ \alpha &\longmapsto \alpha^2. \end{aligned}$$

Since $-\alpha$ and α are distinct elements of $\mathbb{F}_p$ if $\alpha \neq 0$ and $p \neq 2$, and because $(-\alpha)^2 = \alpha$, we see that the mapping $\alpha \longmapsto \alpha^2$ is exactly two-to-one on the non-zero elements of $\mathbb{F}_p$. So if an element x in $\mathbb{F}_p$ has a square root, then it has exactly two square roots (unless $x = 0$) and exactly half of the elements of $\mathbb{F}_p^*$ are squares. The set of squares in $\mathbb{F}_p^*$ are called the *quadratic residues* and they form a subgroup of order $(p-1)/2$ of the multiplicative group $\mathbb{F}_p^*$. The elements of $\mathbb{F}_p^*$ which are not squares are called the *quadratic non-residues*.

To make it easy to detect squares modulo a prime p we define the *Legendre symbol*

$$\left(\frac{a}{p}\right).$$

This is defined to be equal to 0 if p divides a, equal to $+1$ if a is a quadratic residue and equal to -1 if a is a quadratic non-residue.

Notice that, if $a \neq 0$ is a square then it has order dividing $(p-1)/2$ since there is an s such that $s^2 = a$ and s has order dividing $(p-1)$ (by Lagrange's Theorem). Hence if a is a square it must have order dividing $(p-1)/2$, and so $a^{(p-1)/2} \pmod{p} = 1$. However, if a is not a square then by the same reasoning it cannot have order dividing $(p-1)/2$. We then have that $a^{(p-1)/2} = u$ for some u which will have order 2, and hence $u = -1$. Putting these two facts together implies we can easily compute the Legendre symbol, via

$$\left(\frac{a}{p}\right) = a^{(p-1)/2} \pmod{p}.$$

Using the above formula turns out to be a very inefficient way to compute the Legendre symbol. In practice one uses the *law of quadratic reciprocity*

$$\left(\frac{q}{p}\right) = \left(\frac{p}{q}\right)(-1)^{(p-1)(q-1)/4}. \tag{1}$$

In other words we have

$$\left(\frac{q}{p}\right) = \begin{cases} -\left(\frac{p}{q}\right) & \text{If } p = q = 3 \pmod 4, \\ \left(\frac{p}{q}\right) & \text{Otherwise.} \end{cases}$$

Using this law with the following additional formulae gives rise to a recursive algorithm for the Legendre symbol:

$$\left(\frac{q}{p}\right) = \left(\frac{q \pmod p}{p}\right), \tag{2}$$

$$\left(\frac{q \cdot r}{p}\right) = \left(\frac{q}{p}\right) \cdot \left(\frac{r}{p}\right), \tag{3}$$

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}. \tag{4}$$

Assuming we can factor, we can now compute the Legendre symbol

$$\begin{aligned}
\left(\frac{15}{17}\right) &= \left(\frac{3}{17}\right) \cdot \left(\frac{5}{17}\right) && \text{by equation (3)} \\
&= \left(\frac{17}{3}\right) \cdot \left(\frac{17}{5}\right) && \text{by equation (1)} \\
&= \left(\frac{2}{3}\right) \cdot \left(\frac{2}{5}\right) && \text{by equation (2)} \\
&= (-1) \cdot (-1)^3 && \text{by equation (4)} \\
&= 1.
\end{aligned}$$

In a moment we shall see a more efficient algorithm which does not require us to factor integers.

1.3.6. Computing Square Roots Modulo p: Computing square roots of elements in $\mathbb{F}_p^*$ when the square root exists turns out to be an easy task. Algorithm 1.3 gives one method, called Shanks' Algorithm, of computing the square root of a modulo p, when such a square root exists. When $p = 3 \pmod 4$, instead of the Shank's algorithm, we can use the following formula

$$x \leftarrow a^{(p+1)/4} \pmod p,$$

which has the advantage of being deterministic and more efficient than the general method of Shanks. That this formula works is because

$$x^2 = a^{(p+1)/2} = a^{(p-1)/2} \cdot a = \left(\frac{a}{p}\right) \cdot a = a$$

where the last equality holds since we have assumed that a is a quadratic residue modulo p and so it has Legendre symbol equal to one.

1.3.7. The Jacobi Symbol: The Legendre symbol above is only defined when its denominator is a prime, but there is a generalization to composite denominators called the *Jacobi symbol*. Suppose $n \geq 3$ is odd and

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k},$$

then the Jacobi symbol

$$\left(\frac{a}{n}\right)$$

is defined in terms of the Legendre symbol by

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}.$$

The Jacobi symbol can be computed using a similar method to the Legendre symbol by making use of the identity, derived from the law of quadratic reciprocity,

$$\left(\frac{a}{n}\right) = \left(\frac{2}{n}\right)^e \cdot \left(\frac{n \pmod{a_1}}{a_1}\right) (-1)^{(a_1-1)\cdot(n-1)/4}.$$

Algorithm 1.3: Shanks' algorithm for extracting a square root of a modulo p

Choose a random n until one is found such that

$$\left(\frac{n}{p}\right) = -1.$$

Let e, q be integers such that q is odd and $p - 1 = 2^e \cdot q$.
$y \leftarrow n^q \pmod{p}$.
$r \leftarrow e$.
$x \leftarrow a^{(q-1)/2} \pmod{p}$.
$b \leftarrow a \cdot x^2 \pmod{p}$.
$x \leftarrow a \cdot x \pmod{p}$.
while $b \neq 1 \pmod{p}$ **do**
  Find the smallest m such that $b^{2^m} = 1 \pmod{p}$.
  $t \leftarrow y^{2^{r-m-1}} \pmod{p}$.
  $y \leftarrow t^2 \pmod{p}$.
  $r \leftarrow m$.
  $x \leftarrow x \cdot t \pmod{p}$.
  $b \leftarrow b \cdot y \pmod{p}$.
return x.

where $a = 2^e \cdot a_1$ and a_1 is odd. We also have the identities, for n odd,

$$\left(\frac{1}{n}\right) = 1, \ \left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}, \ \left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}.$$

This now gives us a fast algorithm, which does not require factoring of integers, to determine the Jacobi symbol, and so the Legendre symbol in the case where the denominator is prime. The only factoring required is to extract the even part of a number. See Algorithm 1.4 which computes the symbol $\left(\frac{a}{b}\right)$. As an example we have

$$\left(\frac{15}{17}\right) = (-1)^{56}\left(\frac{17}{15}\right) = \left(\frac{2}{15}\right) = (-1)^{28} = 1.$$

1.3.8. Squares and Pseudo-squares Modulo a Composite: Recall that the Legendre symbol $\left(\frac{a}{p}\right)$ tells us whether a is a square modulo p, for p a prime. Alas, the Jacobi symbol $\left(\frac{a}{n}\right)$ does not tell us the whole story about whether a is a square modulo n, when n is a composite. If a is a square modulo n then the Jacobi symbol will be equal to plus one, however if the Jacobi symbol is equal to plus one then it is not always true that a is a square.

Let $n \geq 3$ be odd and let the set of squares in $(\mathbb{Z}/n\mathbb{Z})^*$ be denoted by

$$Q_n = \{x^2 \pmod{n} : x \in (\mathbb{Z}/n\mathbb{Z})^*\}.$$

Now let J_n denote the set of elements with Jacobi symbol equal to plus one, i.e.

$$J_n = \left\{x \in (\mathbb{Z}/n\mathbb{Z})^* : \left(\frac{a}{n}\right) = 1\right\}.$$

The set of pseudo-squares is the difference $J_n \setminus Q_n$. There are two important cases for cryptography, either n is prime or n is the product of two primes:

- n is a prime p:
  - $Q_n = J_n$.
  - $\#Q_n = (n-1)/2$.

Algorithm 1.4: Jacobi symbol algorithm

```
if b ≤ 0 or b (mod 2) = 0 then return 0.
j ← 1.
if a < 0 then
    a ← −a.
    if b (mod 4) = 3 then j ← −j.
while a ≠ 0 do
    while a (mod 2) = 0 do
        a ← a/2.
        if b (mod 8) = 3 or b (mod 8) = 5 then j ← −j.
    (a, b) ← (b, a).
    if a (mod 4) = 3 and b (mod 4) = 3 then j ← −j.
    a ← a (mod b).
if b = 1 then return j.
return 0.
```

- n is the product of two primes, $n = p \cdot q$:
  - $Q_n \subset J_n$.
  - $\#Q_n = \#(J_n \setminus Q_n) = (p-1)(q-1)/4$.

The sets Q_n and J_n will be seen to be important in a number of algorithms and protocols, especially in the case where n is a product of two primes.

1.3.9. Square Roots Modulo $n = p \cdot q$: We now look at how to compute a square root modulo a composite number $n = p \cdot q$. Suppose we wish to compute the square root of a modulo n. We assume we know p and q, and that a really is a square modulo n, which can be checked by demonstrating that

$$\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1.$$

We first compute a square root of a modulo p, call this s_p. Then we compute a square root of a modulo q, call this s_q. Finally to deduce the square root modulo n, we apply the Chinese Remainder Theorem to the equations

$$x = s_p \pmod{p} \text{ and } x = s_q \pmod{q}.$$

Note that if we do not know the prime factors of n then computing square roots modulo n is believed to be a very hard problem; indeed it is as hard as factoring n itself.

As an example suppose we wish to compute the square root of $a = 217$ modulo $n = 221 = 13 \cdot 17$. Now a square root of a modulo 13 and 17 is given by

$$s_{13} = 3 \text{ and } s_{17} = 8.$$

Applying the Chinese Remainder Theorem we find

$$s = 42$$

and we can check that s really is a square root by computing

$$s^2 = 42^2 = 217 \pmod{n}.$$

There are three other square roots, since n has two prime factors. These other square roots are obtained by applying the Chinese Remainder Theorem to the other three equation pairs

$$\begin{aligned} s_{13} &= 10, \quad s_{17} = 8, \\ s_{13} &= 3, \quad s_{17} = 9, \\ s_{13} &= 10, \quad s_{17} = 9, \end{aligned}$$

Hence, all four square roots of 217 modulo 221 are given by 42, 94, 127 and 179.

1.4. Probability

At some points we will need a basic understanding of elementary probability theory. In this section we summarize the theory we require and give a few examples. Most readers should find this a revision of the type of probability encountered in high school. A *random variable* is a variable X which takes certain values with given probabilities. If X takes the value s with probability 0.01 we write this as

$$p(X = s) = 0.01.$$

As an example, let T be the random variable representing tosses of a fair coin, we then have the probabilities

$$\begin{aligned} p(T = \text{Heads}) &= \frac{1}{2}, \\ p(T = \text{Tails}) &= \frac{1}{2}. \end{aligned}$$

As another example let E be the random variable representing letters in English text. An analysis of a large amount of English text allows us to approximate the relevant probabilities by

$$\begin{aligned} p(E = a) &= 0.082, \\ &\vdots \\ p(E = e) &= 0.127, \\ &\vdots \\ p(E = z) &= 0.001. \end{aligned}$$

Basically if X is a discrete random variable on a set S, and $p(X = x)$ is the *probability distribution*, i.e. the probability of a value x being selected from S, then we have the two following properties:

$$\begin{aligned} p(X = x) &\geq 0 \text{ for all } x \in S, \\ \sum_{x \in S} p(X = x) &= 1. \end{aligned}$$

It is common to illustrate examples from probability theory using a standard deck of cards. We shall do likewise and let V denote the random variable that a card is a particular value, let S denote the random variable that a card is a particular suit and let C denote the random variable of the colour of a card. So for example

$$\begin{aligned} p(C = \text{Red}) &= \frac{1}{2}, \\ p(V = \text{Ace of Clubs}) &= \frac{1}{52}, \\ p(S = \text{Clubs}) &= \frac{1}{4}. \end{aligned}$$

Let X and Y be two random variables, where $p(X = x)$ is the probability that X takes the value x and $p(Y = y)$ is the probability that Y takes the value y. The *joint probability* $p(X = x, Y = y)$ is defined as the probability that X takes the value x and Y takes the value y. So if we let $X = C$ and $Y = S$ then we have

$$\begin{aligned} &p(C = \text{Red}, S = \text{Club}) = 0, && p(C = \text{Red}, S = \text{Diamonds}) = \frac{1}{4}, \\ &p(C = \text{Red}, S = \text{Hearts}) = \frac{1}{4}, && p(C = \text{Red}, S = \text{Spades}) = 0, \\ &p(C = \text{Black}, S = \text{Club}) = \frac{1}{4}, && p(C = \text{Black}, S = \text{Diamonds}) = 0, \\ &p(C = \text{Black}, S = \text{Hearts}) = 0, && p(C = \text{Black}, S = \text{Spades}) = \frac{1}{4}. \end{aligned}$$

Two random variables X and Y are said to be *independent* if, for all values of x and y,

$$p(X = x, Y = y) = p(X = x) \cdot p(Y = y).$$

Hence, the random variables C and S are not independent. As an example of independent random variables consider the two random variables T_1 the value of the first toss of an unbiased coin and T_2 the value of a second toss of the coin. Since, assuming standard physical laws, the toss of the first coin does not affect the outcome of the toss of the second coin, we say that T_1 and T_2 are independent. This is confirmed by the joint probability distribution

$$\begin{aligned} &p(T_1 = H, T_2 = H) = \frac{1}{4}, && p(T_1 = H, T_2 = T) = \frac{1}{4}, \\ &p(T_1 = T, T_2 = H) = \frac{1}{4}, && p(T_1 = T, T_2 = T) = \frac{1}{4}. \end{aligned}$$

1.4.1. Bayes' Theorem: The *conditional probability* $p(X = x \mid Y = y)$ of two random variables X and Y is defined as the probability that X takes the value x given that Y takes the value y. Returning to our random variables based on a pack of cards we have

$$p(S = \text{Spades} \mid C = \text{Red}) = 0$$

and

$$p(V = \text{Ace of Spades} \mid C = \text{Black}) = \frac{1}{26}.$$

The first follows since if we know that a card is red, then the probability that it is a spade is zero, since a red card cannot be a spade. The second follows since if we know a card is black then we have restricted the set of cards to half the pack, one of which is the ace of spades.

The following is one of the most crucial statements in probability theory, which you should recall from high school,

Theorem 1.7 (Bayes' Theorem)**.** *If* $p(Y = y) > 0$ *then*

$$\begin{aligned} p(X = x | Y = y) &= \frac{p(X = x) \cdot p(Y = y \mid X = x)}{p(Y = y)} \\ &= \frac{p(X = x, Y = y)}{p(Y = y)}. \end{aligned}$$

We can apply Bayes' Theorem to our examples above as follows

$$\begin{aligned} p(S = \text{Spades} \mid C = \text{Red}) &= \frac{p(S = \text{Spades}, C = \text{Red})}{p(C = \text{Red})} \\ &= 0 \cdot \left(\frac{1}{4}\right)^{-1} = 0. \end{aligned}$$

$$\begin{aligned} p(V = \text{Ace of Spades} \mid C = \text{Black}) &= \frac{p(V = \text{Ace of Spades}, C = \text{Black})}{p(C = \text{Black})} \\ &= \frac{1}{52} \cdot \left(\frac{1}{2}\right)^{-1} \\ &= \frac{2}{52} = \frac{1}{26}. \end{aligned}$$

If X and Y are independent then we have

$$p(X = x|Y = y) = p(X = x),$$

i.e. the value that X takes does not depend on the value that Y takes. An identity which we will use a lot is the following, for events A and B

$$\begin{aligned} p(A) &= p(A, B) + p(A, \neg B) \\ &= p(A|B) \cdot p(B) + p(A|\neg B) \cdot p(\neg B). \end{aligned}$$

where $\neg B$ is the event that B does not happen.

1.4.2. Birthday Paradox: Another useful result from elementary probability theory that we will require is the *birthday paradox*. Suppose a bag has m balls in it, all of different colours. We draw one ball at a time from the bag and write down its colour, we then replace the ball in the bag and draw again. If we define

$$m^{(n)} = m \cdot (m-1) \cdot (m-2) \cdots (m-n+1)$$

then the probability, after n balls have been taken out of the bag, that we have obtained at least one matching colour (or coincidence) is

$$1 - \frac{m^{(n)}}{m^n}.$$

As m becomes larger the expected number of balls we have to draw before we obtain the first coincidence is

$$\sqrt{\frac{\pi \cdot m}{2}}.$$

To see why this is called the birthday paradox consider the probability of two people in a room sharing the same birthday. Most people initially think that this probability should be quite low, since they are thinking of the probability that someone in the room shares the same birthday as them. One can now easily compute that the probability of at least two people in a room of 23 people having the same birthday is

$$1 - \frac{365^{(23)}}{365^{23}} \approx 0.507.$$

In fact this probability increases quite quickly since in a room of 30 people we obtain a probability of approximately 0.706, and in a room of 100 people we obtain a probability of over 0.999 999 6.

In many situations in cryptography we use the birthday paradox in the following way. We are given a random process which outputs elements from a set of size m, just like the balls above. We run the process for n steps, again just like above. But instead of wanting to know how many times we need to execute the process to find a collision we instead want to know an upper bound on the probability of finding a collision after n steps (think of n being much smaller than m). This is easy

to estimate due to the following inequalities:

$$\begin{aligned}\Pr\left[\text{At least one repetition in pulling } n \text{ elements from } m\right] &\leq \sum_{1\leq i<j<n} \Pr[\text{Item } i \text{ collides with item } j] \\ &= \binom{n}{2} \cdot \frac{1}{m} \\ &= \frac{n\cdot(n-1)}{m} \leq \frac{n^2}{2\cdot m}.\end{aligned}$$

1.5. Big Numbers

At various points we need to discuss how big a number can be before it is impossible for someone to perform that many operations. Such big numbers are used in cryptography to measure the work effort of the adversary. Suppose we had a (mythical) computer which could do one trillion "basic" operations per second. Note that a modern 3 GHz computer with eight "cores" can only do 24 billion operations per second, so our mythical computer is around 42 times faster than a current desktop computer.

Suppose we had an algorithm which took 2^t "basic" operations. We want to know how long our mythical computer would take to perform these 2^t operations. Now one trillion is about 2^{40}. Thus to perform 2^{64} operations would require $2^{64-40} = 2^{24}$ seconds, or 194 days. Given that finding 194 computers is not very hard, a calculation which takes 2^{64} basic operations could be performed by someone with just under 200 computers in under a day. An algorithm which took 2^{80} "basic" operations would take 2^{40} seconds for our mythical computer, or nearly 34 900 years. Thus a large government-funded laboratory which could afford perhaps 15 000 mythical computers could perform the algorithm requiring 2^{80} operations in about two years. This might be expensive, but if national security depended on it, then a computation of 2^{80} operations would be plausible.

However, when we go to an algorithm which requires 2^{128} operations then our mythical computer would require 2^{88} seconds or 9 quintillion years (i.e. $9{\cdot}10^{18}$ years). Note, the universe is only believed to be 13.8 billion years old. Thus a computation which required 9 quintillion years is essentially impossible, ever!!!!

To get an idea of how big these numbers are consider that 2^{80} is a number with 24 decimal digits, whereas 2^{128} is a number with 38 decimal digits. These are both significantly more than the number of cells in the human body (which is around 10^{14}), or the number of stars in the observable universe (which is around 10^{22}).

Chapter Summary

- A group is a set with an operation which has an identity, is associative and in which every element has an inverse.
- Addition and multiplication in modular arithmetic both provide examples of groups.
- For modular multiplication we need to be careful which set of numbers we take when defining such a group, as not all integers modulo m are invertible with respect to multiplication.

- A ring is a set with two operations which behaves like the set of integers under addition and multiplication. Modular arithmetic is an example of a ring.
- A field is a ring in which all non-zero elements have a multiplicative inverse. The integers modulo a prime is an example of a field.
- Multiplicative inverses for modular arithmetic can be found using the extended Euclidean algorithm.
- Sets of simultaneous linear modular equations can be solved using the Chinese Remainder Theorem.
- Square elements modulo a prime can be detected using the Legendre symbol; square roots can be efficiently computed using Shanks' Algorithm.
- Square elements and square roots modulo a composite can be determined efficiently as long as one knows the factorization of the modulus.
- Bayes' Theorem allows us to compute conditional probabilities.
- The birthday paradox allows us to estimate how quickly collisions occur when one repeatedly samples from a finite space.
- We also discussed how big various numbers are, as a means to work out what is a feasible computation.

Further Reading

Bach and Shallit is the best introductory book I know of which deals with Euclid's algorithm and finite fields. It contains a lot of historical information, plus excellent pointers to the relevant research literature. Whilst aimed in some respects at Computer Scientists, Bach and Shallit's book may be a little too mathematical for some. For a more traditional introduction to the basic discrete mathematics we shall need, see the books by Biggs or Rosen.

E. Bach and J. Shallit. *Algorithmic Number Theory. Volume 1: Efficient Algorithms*. MIT Press, 1996.

N.L. Biggs. *Discrete Mathematics*. Oxford University Press, 1989.

K.H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill, 1999.