Chapter 7

AN INTRODUCTION TO THE MATHEMATICS OF CODING AND CRYPTOGRAPHY

7.1 Introduction

Shannon's demonstration in 1948 by means of "random coding" arguments that there exist codes that can provide reliable communications over any noisy channel at any rate less than its capacity touched off an immediate and extensive search for specific such codes that continues unabated today. Our purpose in this chapter is to introduce the mathematics, a mixture of number theory and algebra, that has proved most useful in the construction of good channel codes. We shall see later that this same mathematics is equally useful in cryptography (or "secrecy coding"). Channel coding and secrecy coding are closely related. The goal in channel coding is to transmit a message in such a form "that it cannot be misunderstood by the intended receiver"; the goal in secrecy coding includes the additional proviso "but that it cannot be understood by anyone else".

"Coding" in its most general sense refers to any transformation of information from one form to another. Information theory deals with three quite distinct forms of coding: source coding (or "data compression"), channel coding, and secrecy coding. It has become accepted terminology, however, to use the term "coding" with no further qualifier to mean "channel coding", and we have followed this practice in the title of this chapter.

We assume that the reader is familiar with some elementary properties of the integers, but otherwise this chapter is self-contained.

7.2 Euclid's Division Theorem for the Integers

Here and hereafter, we will write $\mathbb{Z}$ to denote the set $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ of *integers*. It is doubtful whether there is any result in mathematics that is more important or more useful than the division theorem for $\mathbb{Z}$ that was stated some 2'300 years ago by Euclid.

Euclid's Division Theorem for the Integers: Given any integers n (the "dividend") and d (the "divisor") with $d \neq 0$, there exist unique integers q (the "quotient") and r (the "remainder") such that

$$n = qd + r \tag{7.1}$$

and

$$0 \leq r < |d|. \tag{7.2}$$

Proof: We show first uniqueness. Suppose that both (q, r) and (q', r') satisfy (7.1) and (7.2). Then $q'd + r' = qd + r$ or, equivalently, $(q' - q)d = r - r'$. But (7.2) implies $|r - r'| < |d|$, whereas $|(q' - q)d| \geq |d|$ if $q' \neq q$. Hence we must have $q' = q$ and thus also $r' = r$.

To show the existence of a pair (q, r) that satisfy (7.1) and (7.2), we suppose without loss of essential generality that $n \geq 0$ and $d > 0$ and consider the decreasing sequence $n, n - d, n - 2d, \ldots$. Let $r = n - qd$ be the last nonnegative term in this sequence so that $r \geq 0$ but $n - (q + 1)d = r - d < 0$ or, equivalently, $0 \leq r < d$. We see now that (7.1) and (7.2) are satisfied; indeed we have described a (very inefficient) algorithm to find q and r. □

Our primary interest hereafter will be in remainders. We will write $R_d(n)$ to denote the *remainder when n is divided by the non-zero integer d.* Because $|d| = |-d|$ and because $qd + r = (-q)(-d) + r$, it follows from Euclid's theorem that

$$R_d(n) = R_{-d}(n) \tag{7.3}$$

for any non-zero integer d. One sees from (7.3) that, with no essential loss of generality, one could restrict one's attention to positive divisors. Later, we shall avail ourselves of this fact. For the present, we will merely adopt the *convention*, whenever we write $R_d(\cdot)$, that d is a *non-zero integer.*

7.3 Properties of Remainders in $\mathbb{Z}$

We begin our study of remainders with a very simple but useful fact.

Fundamental Property of Remainders in $\mathbb{Z}$: For any integers n and i,

$$R_d(n + id) = R_d(n), \tag{7.4}$$

i.e., *adding any multiple of the divisor to the dividend does not alter the remainder.*

Proof: Suppose q and r satisfy (7.1) and (7.2) so that $r = R_d(n)$. Then $n + id = qd + r + id = (q + i)d + r$ so that the new quotient is $q + i$ but the remainder $R_d(n + id)$ is still r. □

With the aid of this fundamental property of remainders, we can now easily prove the following two properties that we call "algebraic properties" because of the crucial role that they play in the algebraic systems that we shall soon consider.

Algebraic Properties of Remainders in $\mathbb{Z}$: For any integers n_1 and n_2,

$$R_d(n_1 + n_2) = R_d(R_d(n_1) + R_d(n_2)), \tag{7.5}$$

i.e., *the remainder of a sum is the remainder of the sum of the remainders*, and

$$R_d(n_1 n_2) = R_d(R_d(n_1) R_d(n_2)), \tag{7.6}$$

i.e., *the remainder of a product is the remainder of the product of the remainders.*

Proof: Let q_1 and r_1 be the quotient and remainder, respectively, when n_1 is divided by d. Similarly let q_2 and r_2 be the quotient and remainder, respectively, when n_2 is divided by d. Then (7.1) gives

$$\begin{aligned} n_1 + n_2 &= q_1 d + r_1 + q_2 d + r_2 \\ &= (q_1 + q_2)d + (r_1 + r_2). \end{aligned}$$

It follows now from the fundamental property (7.4) that

$$R_d(n_1 + n_2) = R_d(r_1 + r_2),$$

which is precisely the assertion of (7.5). Similarly,

$$\begin{aligned} n_1 n_2 &= (q_1 d + r_1)(q_2 d + r_2) \\ &= (q_1 q_2 d + r_1 q_2 + r_2 q_1)d + r_1 r_2. \end{aligned}$$

The fundamental property (7.4) now gives

$$R_d(n_1 n_2) = R_d(r_1 r_2),$$

which is exactly the same as (7.6). □

7.4 Greatest Common Divisors in $\mathbb{Z}$

One says that the non-zero integer d *divides* n just in case that $R_d(n) = 0$ or, equivalently, if there is an integer q such that $n = qd$. Note that $n = 0$ is divisible by every non-zero d, and note also that $d = 1$ divides every n. If $n \neq 0$, then $d = |n|$ is the largest integer that divides n, but $n = 0$ has no largest divisor.

If n_1 and n_2 are integers not both 0, then their *greatest common divisor*, which is denoted by $\gcd(n_1, n_2)$, is the largest integer d that divides both n_1 and n_2. We adopt the *convention*, whenever we write $\gcd(n_1, n_2)$, that n_1 and n_2 are *integers not both 0.* From what we said above, we know that $\gcd(n_1, n_2)$ must be at least 1, and can be at most $\min(|n_1|, |n_2|)$ if both n_1 and n_2 are non-zero. We seek a simple way to find $\gcd(n_1, n_2)$. The following fact will be helpful.

Fundamental Property of Greatest Common Divisors in $\mathbb{Z}$: For any integer i,

$$\gcd(n_1 + in_2, n_2) = \gcd(n_1, n_2), \tag{7.7}$$

i.e., *adding a multiple of one integer to the other does not change their greatest common divisor.*

Proof: We will show that *every divisor* of n_1 and n_2 is also a divisor of $n_1 + in_2$ and n_2, and conversely; hence the greatest common divisors of these two pairs of integers must coincide. Suppose first that d divides n_1 and n_2, i.e., that $n_1 = q_1 d$ and $n_2 = q_2 d$. Then $n_1 + in_2 = q_1 d + iq_2 d = (q_1 + iq_2)d$ so that d also divides $n_1 + in_2$ (and of course n_2). Conversely, suppose that d divides $n_1 + in_2$ and n_2, i.e., that $n_1 + in_2 = q_3 d$ and $n_2 = q_2 d$. Then $n_1 = q_3 d - in_2 = q_3 d - iq_2 d = (q_3 - iq_2)d$ so that d also divides n_1 (and of course n_2). □

If $n_2 \neq 0$, then we can choose i in (7.7) as the negative of the quotient q when n_1 is divided by $d = n_2$, which gives $n_1 + in_2 = n_1 - qd = R_d(n_1)$. Thus, we have proved the following recursion, which is the basis for a fast algorithm for computing $\gcd(n_1, n_2)$.

Euclid's Recursion for the Greatest Common Divisor in $\mathbb{Z}$: If n_2 is not 0, then

$$\gcd(n_1, n_2) = \gcd(n_2, R_{n_2}(n_1)). \tag{7.8}$$

To complete an algorithm to obtain $\gcd(n_1, n_2)$ based on this recursion, we need only observe that

$$\gcd(n, 0) = |n|. \tag{7.9}$$

It is convenient to note that negative integers need not be considered when working with greatest common divisors. Because d divides n if and only if d divides $-n$, it follows that

$$\gcd(\pm n_1, \pm n_2) = \gcd(n_1, n_2), \tag{7.10}$$

by which we mean that *any of the four choices for the pair of signs on the left gives the same result.*

We have thus proved the validity of *Euclid's greatest common divisor algorithm*, the flowchart of which is given in Fig. 7.1, which computes $\gcd(n_1, n_2)$ by iterative use of (7.8) until $R_{n_2}(n_1) = 0$ when (7.9) then applies. This algorithm, one of the oldest in mathematics, is nonetheless very efficient and still often used today.

Example 7.4.1 Find $\gcd(132, 108)$. The following table shows the computation made with the algorithm of Fig. 7.1.

n_1	n_2	r	g
132	108	24	-
108	24	12	-
24	12	0	12

Thus, $\gcd(132, 108) = 12$.

We come next to a non-obvious, but exceedingly important, property of the greatest common divisor, namely that *the greatest common divisor of two integers can always be written as an integer combination of the same two integers.*

Greatest Common Divisor Theorem for $\mathbb{Z}$: For any integers n_1 and n_2, not both 0, there exist integers a and b such that

$$\gcd(n_1, n_2) = an_1 + bn_2. \tag{7.11}$$

Remark: The integers a and b specified in this theorem are *not* unique. For instance, $\gcd(15, 10) = 5$ and $5 = (1)(15) + (-1)(10) = (11)(15) + (-16)(10)$. [Note, however, that $R_{10}(1) = R_{10}(11) = 1$ and that $R_{15}(-1) = R_{15}(-16) = 14$.]

Proof: We prove the theorem by showing that the *"extended" Euclidean greatest common divisor algorithm* of Fig. 7.2 computes not only $g = \gcd(n_1, n_2)$ but also a pair of integers a and b such that (7.11) is satisfied. (Because of (7.10), there is no loss of essential generality in assuming that n_2 is positive, as is done in Fig. 7.2.)

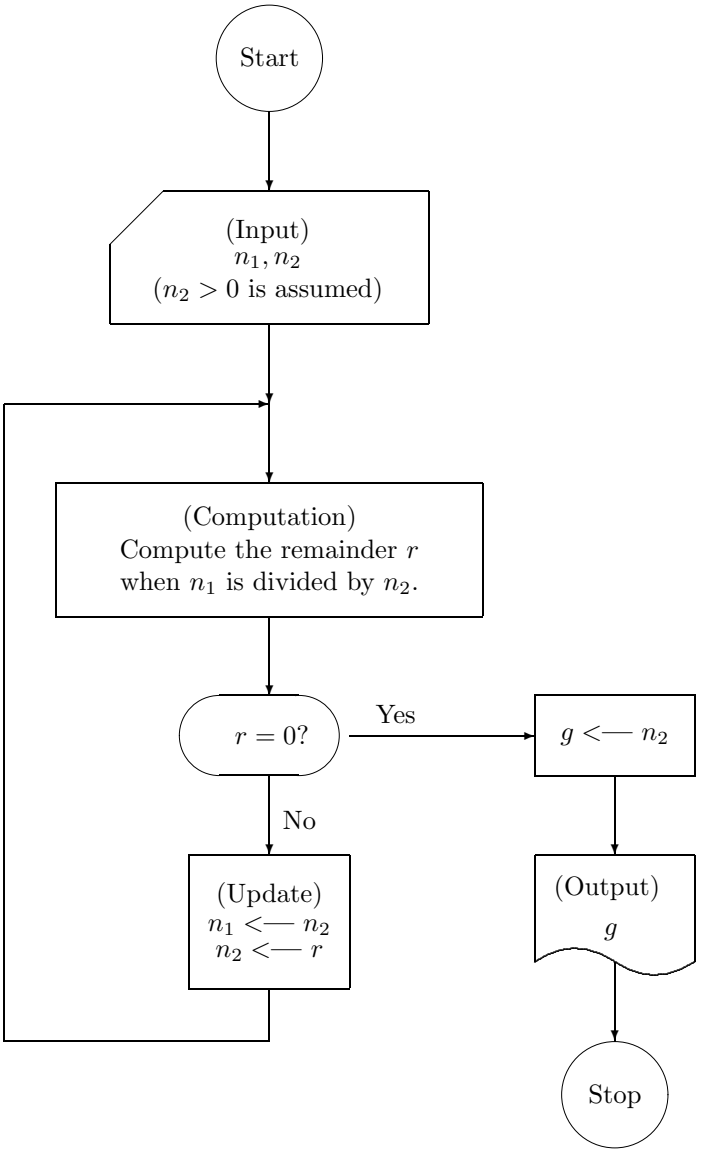

Figure 7.1: Flowchart of Euclid's algorithm for computing the greatest common divisor g of the integers n_1 and n_2 with $n_2 > 0$.

We will write $n_1(i), n_2(i), a_1(i), b_1(i), a_2(i)$ and $b_2(i)$ to denote the values of the variables n_1, n_2, a_1, b_1, a_2 and b_2, respectively, just after the i-th execution of the "update" box in Fig. 7.2; the values for $i = 0$ are the initial values assigned to these variables. Thus, $n_1(0) = n_1, n_2(0) = n_2, a_1(0) = 1, b_1(0) = 0, a_2(0) = 0$ and $b_2(0) = 1$. We claim that the equations

$$n_1(i) = a_1(i)n_1(0) + b_1(i)n_2(0) \tag{7.12}$$

and

$$n_2(i) = a_2(i)n_1(0) + b_2(i)n_2(0) \tag{7.13}$$

are satisfied for all $i \geq 0$ until the algorithm halts. If so, because the final value of n_2 is $\gcd(n_1(0), n_2(0))$, the final value of a_2 and b_2 will be the desired pair of integers a and b satisfying (7.11).

A trivial check shows that (7.12) and (7.13) indeed hold for $i = 0$. We assume now that they hold up to a general i and wish to show they hold also for $i + 1$. But $n_1(i + 1) = n_2(i)$ so that the choices $a_1(i+1) = a_2(i)$ and $b_1(i+1) = b_2(i)$ made by the algorithm guarantee that (7.12) holds with i increased to $i + 1$. Letting $q(i)$ and $r(i)$ be the quotient and remainder, respectively, when $n_1(i)$ is divided by $n_2(i)$, we have

$$n_1(i) = q(i)n_2(i) + r(i). \tag{7.14}$$

But $n_2(i + 1) = r(i)$ and hence

$$\begin{aligned} n_2(i + 1) &= n_1(i) - q(i)n_2(i) \\ &= [a_1(i) - q(i)a_2(i)]n_1(0) + [b_1(i) - q(i)b_2(i)]n_2(0), \end{aligned}$$

where we have made use of (7.12) and (7.13). Thus, the choices $a_2(i + 1) = a_1(i) - q(i)a_2(i)$ and $b_2(i + 1) = b_1(i) - q(i)b_2(i)$ made by the algorithm guarantee that (7.13) also holds with i increased to $i + 1$. □

Example 7.4.2 Find $\gcd(132, 108)$ together with integers a and b such that $\gcd(132, 108) = a(132) + b(108)$. The following table shows the computation made with the algorithm of Fig. 7.2.

n_1	n_2	a_1	b_1	a_2	b_2	q	r	g	a	b
132	108	1	0	0	1	1	24	-	-	-
108	24	0	1	1	-1	4	12	-	-	-
24	12	1	-1	-4	5	2	0	12	-4	5

Thus, $g = \gcd(132, 108) = 12 = a(132) + b(108) = (-4)(132) + (5)(108)$.

Particularly for applications in cryptography, it is important to determine the amount of computation required by the (extended) Euclidean algorithm. Because the main computation in the algorithm is the required integer division, we will count only the number of divisions used. To determine the *worst-case computation*, we define $m(d)$ to be the *minimum value of* n_1 *such that the algorithm uses* d *divisions* for some n_2 satisfying $n_1 > n_2 > 0$. The following values of $m(d)$ and the corresponding values of n_2 are easily determined by inspection of the flowchart in Fig. 7.1 (or Fig. 7.2).

d	$n_1 = m(d)$	n_2
1	2	1
2	3	2
3	5	3

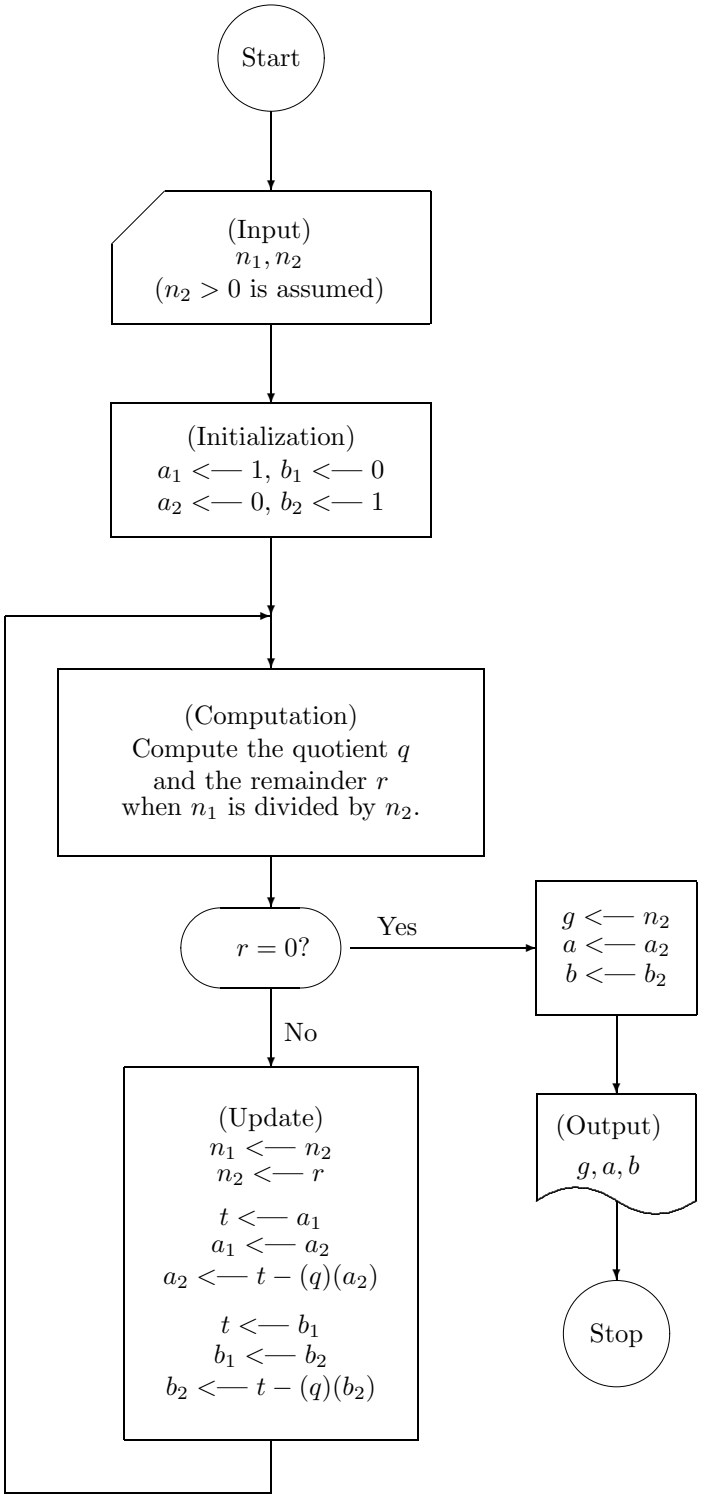

Figure 7.2: Flowchart of the extended Euclidean algorithm for computing $g = \gcd(n_1, n_2)$ together with a pair of integers a and b such that $g = an_1 + bn_2$ for integers n_1 and n_2 with $n_2 > 0$.

Because $n_1(i+1) = n_2(i)$ and $n_1(i+2) = n_2(i+1) = r(i)$, we see from (7.14) that

$$n_1(i) = q(i)n_1(i+1) + n_1(i+2).$$

If $n_1(i) = m(d)$, then we must have $n_1(i+1) \geq m(d-1)$ and $n_1(i+2) \geq m(d-2)$ and hence [since we will always have $q(i) \geq 1$ when we begin with $n_1 > n_2 > 0$] we must have $m(d) \geq m(d-1) + m(d-2)$. But in fact equality holds because equality will always give $q(i) = 1$. Thus, the worst-case computation satisfies

$$m(d) = m(d-1) + m(d-2), \tag{7.15}$$

which is just *Fibonacci's recursion*, with the initial conditions

$$m(1) = 2,\ m(2) = 3. \tag{7.16}$$

Solving this recursion gives

$$m(d) = \left(\frac{3\sqrt{5}+5}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^d - \left(\frac{3\sqrt{5}-5}{10}\right)\left(\frac{1-\sqrt{5}}{2}\right)^d. \tag{7.17}$$

A simple check shows that the second term on the right of (7.17) has a magnitude at most .106 for $d \geq 1$. Thus, to a very good approximation,

$$m(d) \approx \left(\frac{3\sqrt{5}+5}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^d$$

or, equivalently,

$$\log_2(m(d)) \approx .228 + .694d$$

or, again equivalently,

$$d \approx 1.44\log_2(m(d)) - .328.$$

We summarize these facts as follows.

Worst-Case Computation for the Euclidean Algorithm: When $n_1 > n_2 > 0$, the number d of divisions used by the (extended) Euclidean algorithm to compute $\gcd(n_1, n_2)$ satisfies

$$d < 1.44\log_2(n_1), \tag{7.18}$$

with near equality when n_2 and n_1 are consecutive Fibonacci numbers.

Example 7.4.3 Suppose that n_1 is a number about 100 decimal digits long, i.e., $n_1 \approx 10^{100} \approx 2^{332}$. Then (7.18) shows that

$$d < (1.44)(332) \approx 478$$

divisions will be required to compute $\gcd(n_1, n_2)$ by Euclid's algorithm when $n_1 > n_2 > 0$. In other words, the required number of *divisions* is at most 44% greater than the length of n_1 in binary digits, which is the way that one should read inequality (7.18).

We have no real use for further properties of greatest common divisors, but it would be a shame not to mention here that, after 23 centuries of computational supremacy, Euclid's venerable algorithm was dethroned in 1961 by an algorithm due to Stein. Stein's approach, which is the essence of simplicity, rests on the following facts:

Even/Odd Relationships for Greatest Common Divisors:

(i) If n_1 and n_2 are *both even* (and not both 0), then

$$\gcd(n_1, n_2) = 2\gcd(n_1/2, n_2/2). \tag{7.19}$$

(ii) If n_1 is *even* and n_2 is *odd*, then

$$\gcd(n_1, n_2) = \gcd(n_1/2, n_2). \tag{7.20}$$

(iii) If n_1 and n_2 are *both odd*, then

$$\gcd(n_1, n_2) = \gcd((n_1 - n_2)/2, n_2). \tag{7.21}$$

Proof: The relationships (i) and (ii) are self-evident. To prove (iii), we suppose that n_1 and n_2 are both odd. From the fundamental property (7.7), it follows that $\gcd(n_1, n_2) = \gcd(n_1 - n_2, n_2)$. But $n_1 - n_2$ is even and n_2 is odd so that (ii) now implies (iii). □

The three even/odd relationships immediately lead to *Stein's greatest common divisor algorithm*, the flowchart of which is given in Fig. 7.3. One merely removes as many common factors of 2, say c, as possible from both n_1 and n_2 according to relationship (i). One then exploits (ii) until the new n_1 and n_2 are both odd. One then applies (iii), returning again to (ii) if $n_1 - n_2 \neq 0$.

Example 7.4.4 Find $\gcd(132, 108)$. The following table shows the computation made with the algorithm of Fig. 7.3.

n_1	n_2	c	g
132	108	0	-
66	54	1	-
33	27	2	-
3	”	”	-
27	3	”	-
12	”	”	-
6	”	”	-
3	”	”	-
0	”	”	12

Note that only 3 *subtractions* (33 - 27, 27 - 3 and 3 - 3) were performed in computing $g = \gcd(132, 108) = 12$. Recall that computing $\gcd(132, 108)$ by Euclid's algorithm in Example 7.4.1 required 3 *divisions*.

We now determine the *worst-case computation* for Stein's algorithm. Note that a test for evenness is just a test on the least significant bit of a number written in radix-two form, and that division of an even number by 2 is just a right shift. Thus, the main computation in the algorithm is the required integer subtraction of n_2 from n_1. We define $\mu(s)$ to be the *minimum value of* $n_1 + n_2$ *such that Stein's algorithm uses* s *subtractions* for integers n_1 and n_2 satisfying $n_1 \geq n_2 > 0$. The following values of $\mu(s)$ and the corresponding values of n_1 and n_2 are easily determined by inspection of the flowchart in Fig. 7.3.

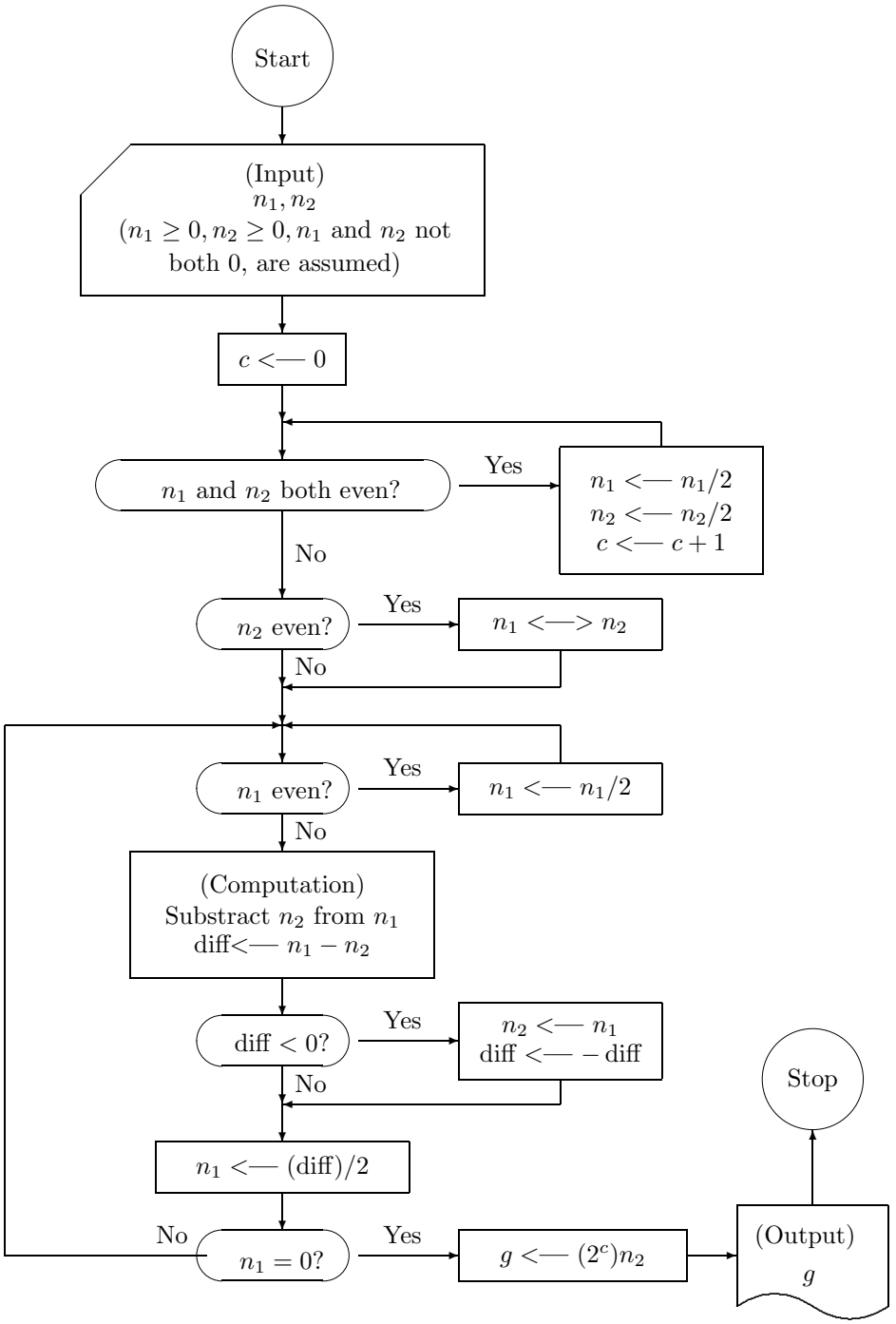

Figure 7.3: Flowchart of Stein's algorithm for computing the greatest common divisor g of the nonnegative integers n_1 and n_2.

s	$\mu(s)$	n_1	n_2
1	2	1	1
2	4	3	1
3	8	5	3
4	16	11	5

After applying relationship (iii), one sees that the new values $n_2' = n_2$ and $n_1' = (n1 - n2)/2$ satisfy

$$n_1' + n_2' = (n_1 - n_2)/2 + n_2 = (n_1 + n_2)/2,$$

i.e., the sum of n_1 and n_2 is reduced by a factor of exactly two when a subtraction is performed. It follows that

$$\mu(s) = 2^s \tag{7.22}$$

for all $s \geq 1$ or, equivalently, that $\log_2(\mu(s)) = s$. Thus, if $n_1 > n_2 > 0$, at most $\log_2(n_1 + n_2) < \log_2(2n_1) = \log_2(n1) + 1$ subtractions are required in Stein's algorithm.

Worst-Case Computation for Stein's Algorithm: When $n_1 > n_2 > 0$, the number s of *subtractions* used by Stein's algorithm to compute $\gcd(n_1, n_2)$ satisfies

$$s < \log_2(n_1) + 1, \tag{7.23}$$

with near equality when n_2 and n_1 are successive integers of the form $(2^{i+1} - (-1)^i)/3$.

Proof: We have already proved everything except the claim about near equality. If we define

$$t_i = \frac{2^{i+1} - (-1)^i}{3}, \tag{7.24}$$

then we find that $(t_i - t_{i-1})/2 = t_{i-2}$ holds for all $i \geq 3$ and, moreover, that $t_i + t_{i-1} = 2^i$. Thus, $n_1 = t_s$ and $n_2 = t_{s-1}$ are indeed the worst-case values of n_1 and n_2 corresponding to $n_1 + n_2 = \mu(s) = 2^s$. □

Example 7.4.5 Suppose that n_1 is a number about 100 decimal digits long, i.e., $n_1 \approx 10^{100} \approx 2^{332}$. Then (7.23) shows that

$$s < 332 + 1 = 333$$

subtractions will be required to compute $\gcd(n_1, n_2)$ by Stein's algorithm when $n_1 > n_2 > 0$.

Comparison of (7.18) and (7.23) shows that the (worst-case) number of *subtractions* required by Stein's algorithm is some 30% less than the (worst-case) number of *divisions* (a much more complex operation!) required by Euclid's algorithm. In fairness to Euclid, however, it must be remembered that in the "worst case" for Euclid's algorithm all quotients are 1 so that the divisions in fact are then subtractions. Nonetheless, one sees that Stein's algorithm is significantly faster than Euclid's and would always be preferred for machine computation. Euclid's algorithm has a certain elegance and conceptual simplicity that alone would justify its treatment here, but there is an even better reason for not entirely abandoning Euclid. Euclid's algorithm generalizes directly to polynomials, as we shall see later, whereas Stein's algorithm is limited to integers.

It should now hardly be surprising to the reader to hear that Stein's greatest common divisor algorithm for computing $\gcd(n_1, n_2)$ can be "extended" to find also integers a and b such that

$\gcd(n_1, n_2) = an_1 + bn_2$. An appropriate extension is given in the flowchart of Fig. 7.4. The basic "trick" is the same as that used to extend Euclid's algorithm. In this case, one ensures that

$$n_1(i) = a_1(i)N_1 + b_1(i)N_2 \tag{7.25}$$

and

$$n_2(i) = a_2(i)N_1 + b_2(i)N_2 \tag{7.26}$$

are satisfied by the initialization and each subsequent updating, where N_1 and N_2 are "n_1" and "n_2" after removal of as many common factors of 2 as possible and where, by choice, N_2 is odd. [The "flag" in the flowchart of Fig. 7.4 reflects this choice of N_2 as odd.] The only non-obvious step in the updating arises when $n_1(i)$ is even, and hence must be divided by 2, but $a_1(i)$ in (7.25) is odd and thus cannot be divided by 2. This problem is resolved by rewriting (7.25) as

$$n_1(i) = [a_1(i) \pm N_2]N_1 + [b_1(i) \mp N_1]N_2 \tag{7.27}$$

where the choice of sign, which is optional, is made so as to keep a_1 and b_1 small in magnitude, which is generally desired in an extended greatest common divisor algorithm. Because $a_1(i) \pm N_2$ is even, the division by 2 can now be performed.

7.5 Semigroups and Monoids

We now begin our treatment of various algebraic systems. In general, an *algebraic system* consists of one or more sets and one or more operations on elements of these sets, together with the axioms that these operations must satisfy. Very often we will consider familiar sets, such as $\mathbb{Z}$, and familiar operations, such as $+$ (addition of integers), where we may already know many additional rules that the operations satisfy. The main task of algebra, when considering a particular type of algebraic system, is to determine those properties that can be proved to be consequences only of the axioms. In this way, one establishes properties that hold for *every* algebraic system of this type regardless of whether or not, in a given system of this type, the operations also satisfy further rules that are not consequences of the axioms. Algebra leads not only to a great economy of thought, but it also greatly deepens one's insight into familiar algebraic systems.

The most elementary algebraic system is the semigroup.

A *semigroup* is an algebraic system $\langle S, * \rangle$, where S is a non-empty set and $*$ is an operation on pairs of elements of S, such that

(A1) (*axiom of closure*) for every a and b in S, $a * b$ is also in S; and

(A2) (*associative law*) for every a, b and c in S, $a * (b * c) = (a * b) * c$.

The axiom of closure (A1) is sometimes only implicitly expressed in the definition of a semigroup by saying that $*$ is a function $*$: $S \times S \to S$ and then writing $a * b$ as the value of the function $*$ when the arguments are a and b. But it is perhaps not wise to bury the axiom of closure in this way; closure is always the axiom that lies at the heart of the definition of an algebraic system.

Example 7.5.1 The system $\langle S, * \rangle$ where S is the set of all binary strings of positive and finite length and where $*$ denotes "string concatenation" (e.g., $10 * 001 = 10001$), is a semigroup.

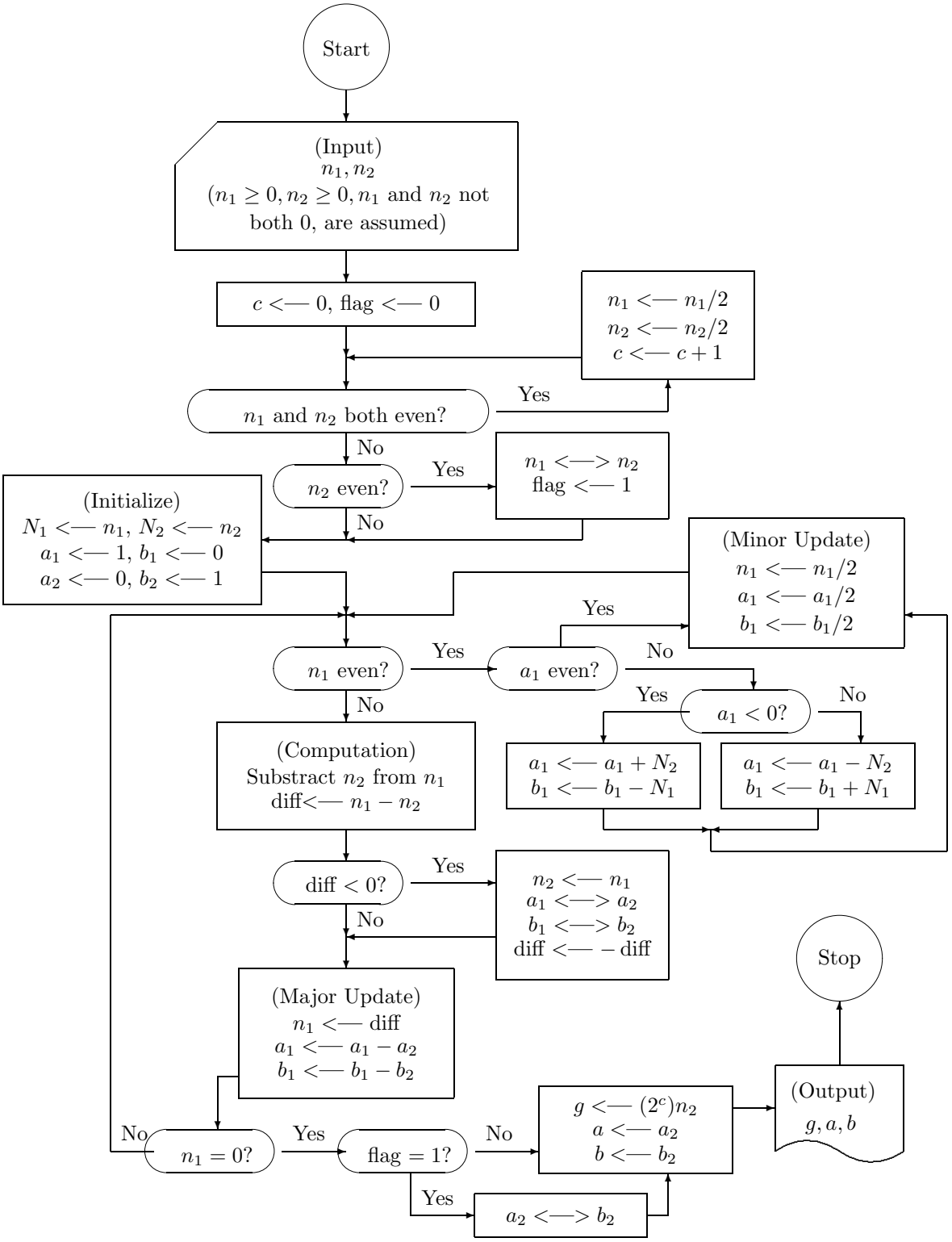

Figure 7.4: Flowchart of the extended Stein algorithm for computing $g = \gcd(n_1, n_2)$ together with integers a and b such that $g = an_1 + bn_2$.

Example 7.5.2 The system $\langle S, * \rangle$, where S is the set of all binary strings of length 3 is *not* a semigroup because axiom (A1) fails. For instance, 101 and 011 are in S, but $101 * 011 = 101011$ is not in S.

Axiom (A2), the associative law, shows that parentheses are not needed to determine the meaning of $a * b * c$; both possible interpretations give the same result. Most of the operations that we use in engineering are associative – with one notable exception, the "vector cross product."

Example 7.5.3 The system $\langle S, \times \rangle$, where S is the set of all vectors in ordinary three-dimensional space and $\times$ denotes the vector cross procuct (which is very useful in dynamics and in electromagnetic theory), is *not* a semigroup because axiom (A2) fails. For example, if $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ are the vectors shown in Fig. 7.5, then $(\mathbf{a} \times \mathbf{b}) \times \mathbf{c} = \mathbf{0}$ but $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) \neq \mathbf{0}$.

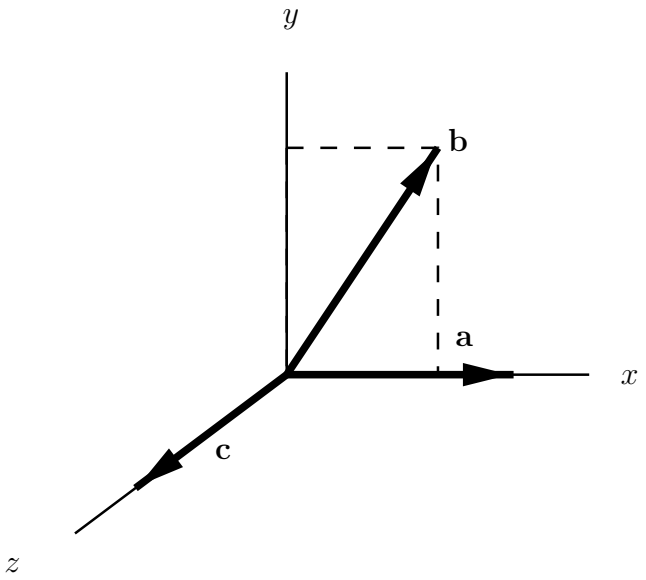

Figure 7.5: Example showing that the vector cross product does not obey the associative law.

A semigroup is almost too elementary an algebraic system to allow one to deduce interesting properties. But the situation is much different if we add one more axiom.

A *monoid* is an algebraic system $\langle M, * \rangle$ such that $\langle M, * \rangle$ is a semigroup and

(A3) (existence of a neutral element) there is an element e of M such that, for every a in M, $a * e = e * a = a$.

Example 7.5.4 The system $\langle \mathbb{Z}, \cdot \rangle$, where $\cdot$ denotes ordinary integer multiplication, is a monoid whose neutral element is $e = 1$.

The semigroup of Example 7.5.1 is *not* a monoid because axiom (A3) fails. [This can be remedied by adding the "empty string" Λ of length 0 to S, in which case $\langle S, * \rangle$ becomes a monoid whose neutral element is $e = \Lambda$.]

We now prove the most important property of monoids.

Uniqueness of the Neutral Element: The neutral element e of a monoid is unique.

Proof: Suppose that e and $\tilde{e}$ are both neutral elements for a monoid $\langle M, * \rangle$. Then

$$\tilde{e} = \tilde{e} * e = e$$

where the first equality holds because e is a neutral element and the second equality holds because $\tilde{e}$ is a neutral element. □

An element c of a monoid $\langle M, * \rangle$ is said to be *invertible* if there is an element c' in M such that $c * c' = c' * c = e$. The element c' is called an *inverse* of c.

Example 7.5.5 In the monoid $\langle \mathbb{Z}, \cdot \rangle$, the only invertible elements are $e = 1$ (where we note that the neutral element e of a monoid is always invertible) and -1. Both of these elements are their own inverses.

Uniqueness of Inverses: If c is an invertible element of the monoid $\langle M, * \rangle$, then its inverse c' is unique.

Proof: Suppose that c' and $\tilde{c}$ are both inverses of c. Then

$$c' = c' * e = c' * (c * \tilde{c}) = (c' * c) * \tilde{c} = e * \tilde{c} = \tilde{c},$$

where the first equality holds because e is the neutral element, the second holds because $\tilde{c}$ is an inverse of c, the third holds because of the associative law (A2), the fourth holds because c' is an inverse of c, and the fifth holds because e is the neutral element. □

Here and hereafter we will write $\mathbb{Z}_m$ to denote the set $\{0, 1, 2, \ldots, m-1\}$ of nonnegative integers less than m, where we shall always assume that *m is at least 2*. Thus, $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ always contains the integers 0 and 1, regardless of the particular choice of m. We now define *multiplication* for the set $\mathbb{Z}_m$ by the rule

$$a \odot b = R_m(a \cdot b) \tag{7.28}$$

where the multiplication on the right is ordinary multiplication of integers. We will call m the *modulus* used for multiplication and we will call the operation $\odot$ *multiplication modulo m*.

The algebraic system $\langle \mathbb{Z}_m, \odot \rangle$ is a monoid whose neutral element is $e = 1$.

Proof: Inequality (7.2) for remainders guarantees that, for every a and b in $\mathbb{Z}_m$, $a \odot b = R_m(a \cdot b)$ is also in $\mathbb{Z}_m$ and thus axiom (A1) is satisfied. For a, b and c in $\mathbb{Z}_m$, we have

$$\begin{aligned}
(a \odot b) \odot c &= R_m((a \odot b) \cdot c) \\
&= R_m(R_m(a \cdot b) \cdot c) \\
&= R_m(R_m(a \cdot b) \cdot R_m(c)) \\
&= R_m((a \cdot b) \cdot c) \\
&= R_m(a \cdot (b \cdot c)) \\
&= R_m(R_m(a) \cdot R_m(b \cdot c)) \\
&= R_m(a \cdot R_m(b \cdot c)) \\
&= R_m(a \cdot (b \odot c)) \\
&= a \odot (b \odot c)
\end{aligned}$$

where the first and second equalities hold because of the definition of $\odot$, the third because c satisfies $0 \leq c < m$, the fourth because of the algebraic property (7.6) of remainders, the fifth because the

associative law holds for ordinary integer multiplication, the sixth again because of the algebraic property (7.6) of remainders, the seventh because a satisfies $0 \leq a < m$, and the eighth and ninth because of the definition (7.28) of $\odot$. Thus axiom (A2) holds in $\langle \mathbb{Z}_m, \odot \rangle$ so that $\langle \mathbb{Z}_m, \odot \rangle$ is a semigroup. But, for any a in $\mathbb{Z}_m$, $a \odot 1 = R_m(a \cdot 1) = R_m(a) = a$ and similarly $1 \odot a = a$. Thus 1 is a neutral element and hence $\langle \mathbb{Z}_m, \odot \rangle$ is indeed a monoid as claimed. □

We now answer the question as to which elements of the monoid $\langle \mathbb{Z}_m, \odot \rangle$ are invertible.

Invertible elements of $\langle \mathbb{Z}_m, \odot \rangle$: An element u in $\mathbb{Z}_m$ is an invertible element of the monoid $\langle \mathbb{Z}_m, \odot \rangle$ if and only if $\gcd(m, u) = 1$ (where of course u is treated as an ordinary integer in computing the greatest common divisor.)

Proof: Suppose that $g = \gcd(m, u) > 1$. Then, for any b with $0 \leq b < m$, we have

$$b \odot u = R_m(b \cdot u) = b \cdot u - q \cdot m$$

where q is the quotient when $b \cdot u$ is divided by m. But $b \cdot u$ is divisible by g and so also is $q \cdot m$. Thus $b \cdot u - q \cdot m$ is divisible by g and hence cannot be 1. Thus, $b \odot u \neq 1$ for all b in $\mathbb{Z}_m$, and hence u cannot be an invertible element of the monoid $\langle \mathbb{Z}_m, \odot \rangle$.

Conversely, suppose that $\gcd(m, u) = 1$. By the greatest common divisor theorem for $\mathbb{Z}$, there exist integers a and b such that $1 = a \cdot m + b \cdot u$. Taking remainders on both sides of this equation gives

$$\begin{aligned} 1 &= R_m(a \cdot m + b \cdot u) \\ &= R_m(b \cdot u) \\ &= R_m(R_m(b) \cdot R_m(u)) \\ &= R_m(R_m(b) \cdot u) \\ &= R_m(b) \odot u \end{aligned}$$

and an entirely similar argument shows that $u \odot R_m(b) = 1$. Thus u is indeed invertible and $u' = R_m(b)$ is the inverse of u. □

Our proof of the condition for invertibility has in fact shown us how to compute inverses in $\langle \mathbb{Z}_m, \odot \rangle$. [Henceforth, because the operation $\odot$ is a kind of "multiplication", we will write the inverse of u as u^{-1} rather than merely as u'.]

Computing Multiplicative Inverses in $\mathbb{Z}_m$: To find the inverse u^{-1} of an invertible element of $\langle \mathbb{Z}_m, \odot \rangle$, use an extended greatest common divisor algorithm to find the integers a and b such that $1 = \gcd(m, u) = a \cdot m + b \cdot u$. Then $u^{-1} = R_m(b)$.

7.6 Euler's Function and the Chinese Remainder Theorem

Euler's totient function (or simply *Euler's Function*) $\varphi(\cdot)$ is the function defined on the positive integers in the manner that

$$\varphi(n) = \#\{i : 0 \leq i < n \text{ and } \gcd(n, i) = 1\}. \tag{7.29}$$

Note that $\varphi(1) = 1$ because $\gcd(1, 0) = 1$ and note that $n = 1$ is the only positive integer for which $i = 0$ contributes to the cardinality of the set on the right in (7.29). The following enumeration is an

immediate consequence of definition (7.29) and of the necessary and sufficient condition for an element u of $\mathbb{Z}_m$ to have a multiplicative inverse.

Count of Invertible Elements in $\langle \mathbb{Z}_m, \odot \rangle$: There are exactly $\varphi(m)$ invertible elements in the monoid $\langle \mathbb{Z}_m, \odot \rangle$.

For any *prime* p, it follows immediately from the definition (7.29) that

$$\varphi(p) = p - 1. \tag{7.30}$$

More generally, if p is a *prime* and e is a *positive integer*, then

$$\varphi(p^e) = \left(1 - \frac{1}{p}\right) \cdot p^e = (p-1) \cdot p^{e-1}. \tag{7.31}$$

To see this, note that the elements of $\mathbb{Z}_{p^e}$ are precisely the integers i that can be written as $i = q \cdot p + r$ where $0 \leq r < p$ and $0 \leq q < p^{e-1}$. Thus, $\gcd(i, p^e) = 1$ if and only if $r \neq 0$. But there are $p-1$ non-zero values of r and there are p^{e-1} choices for q.

To determine $\varphi(m)$ when m has more than one prime factor, it is convenient to introduce the Chinese Remainder Theorem, for which we will later find many other uses.

Let $m_1, m_2, \ldots, m_k$ be positive integers. Their *least common multiple*, denoted $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$ is the smallest positive integer divisible by each of these positive integers. If m_1 and m_2 are positive integers, then $\mathrm{lcm}(m_1, m_2)\gcd(m_1, m_2) = m_1 m_2$, but this simple relationship does not hold for more than two positive integers.

Fundamental Property of Least Common Multiples: Each of the positive integers $m_1, m_2, \ldots, m_k$ divides an integer n if and only if $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$ divides n.

Proof: If $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$ divides n, i.e., if $n = q \cdot \mathrm{lcm}(m_1, m_2, \ldots, m_k)$, then trivially m_i also divides n for $i = 1, 2, \ldots, k$.

Conversely, suppose that m_i divides n for $i = 1, 2, \ldots, k$. By Euclid's division theorem for $\mathbb{Z}$, we can write n as

$$n = q \cdot \mathrm{lcm}(m_1, m_2, \ldots, m_k) + r \tag{7.32}$$

where

$$0 \leq r < \mathrm{lcm}(m_1, m_2, \ldots, m_k). \tag{7.33}$$

But m_i divides both n and $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$ and thus (7.32) implies that m_i divides r for $i = 1, 2, \ldots, k$. Because (7.33) shows that r is nonnegative and less than $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$, the conclusion must be that r is 0, i.e., that $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$ divides n. □

Pairwise Relative Primeness and Least Common Multiples: If the positive integers $m_1, m_2, \ldots, m_k$ are *pairwise relatively prime* (i.e., if $\gcd(m_i, m_j) = 1$ for $1 \leq i < j \leq k$), then

$$\mathrm{lcm}(m_1, m_2, \ldots, m_k) = m_1 \cdot m_2 \cdot \ldots \cdot m_k. \tag{7.34}$$

Proof: If p^e, where p is a prime and e is a positive integer, is a term in the prime factorization of m_i, then this p cannot be a prime factor of m_j for $j \neq i$ because $\gcd(m_i, m_j) = 1$. Hence p^e must also be a factor of $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$. Thus, $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$ cannot be smaller than the product on the right of (7.34) and, since each m_i divides this product, this product must be $\mathrm{lcm}(m_1, m_2, \ldots, m_k)$. □

Henceforth, we shall assume that $m_i \geq 2$ for each i and we shall refer to the integers $m_1, m_2, \ldots, m_k$ as *moduli*.

Chinese Remainder Theorem: Suppose that $m_1, m_2, \ldots, m_k$ are pairwise relatively prime moduli and let $m = m_1 \cdot m_2 \cdot \ldots \cdot m_k$. Then, for any choice of integers $r_1, r_2, \ldots, r_k$ such that $0 \leq r_i < m_i$ for $i = 1, 2, \ldots, k$, there is a unique n in $\mathbb{Z}_m$ such that

$$R_{m_i}(n) = r_i, \quad i = 1, 2, \ldots, k \tag{7.35}$$

[where, of course, n is treated as an ordinary integer in computing the remainder on the left in (7.35)].

Proof: We first show that (7.35) can be satisfied for at most one n in $\mathbb{Z}_m$. For suppose that (7.35) is satisfied for n and $\tilde{n}$, both in $\mathbb{Z}_m$, where $n \geq \tilde{n}$ when n and $\tilde{n}$ are considered as ordinary integers. Then

$$R_{m_i}(n - \tilde{n}) = R_{m_i}(R_{m_i}(n) - R_{m_i}(\tilde{n})) = R_{m_i}(r_i - \tilde{r}_i) = 0$$

for $i = 1, 2, \ldots, k$, where for the second equality we used the algebraic properties of remainders. Thus $n - \tilde{n}$ is divisible by each of the moduli m_i and hence (by the fundamental property of least common multiples) by $\text{lcm}(m_1, m_2, \ldots, m_k)$. But the pairwise relative primeness of the moduli implies $\text{lcm}(m_1, m_2, \ldots, m_k) = m$. Thus $n - \tilde{n}$ (where $0 \leq n - \tilde{n} < m$ by assumption) is divisible by m so that $n - \tilde{n} = 0$ is the only possible conclusion. It follows that (7.35) is indeed satisfied by at most one n in $\mathbb{Z}_m$ or , equivalently, that *each n in $\mathbb{Z}_m$ has a unique corresponding list of remainders* $(r_1, r_2, \ldots, r_k)$ with $0 \leq r_i < m_i$. But there are m elements in $\mathbb{Z}_m$ and only $m_1 \cdot m_2 \cdot \ldots \cdot m_k = m$ possible choices of lists $(r_1, r_2, \ldots, r_k)$ with $0 \leq r_i < m_i$. It follows that every such list must correspond to exactly one element of $\mathbb{Z}_m$. □

The remainders r_i in (7.35) are often called the *residues* of n with respect to the moduli m_i. When the moduli $m_1, m_2, \ldots, m_k$ are pairwise relatively prime, the Chinese Remainder Theorem (*CRT*) specifies a one-to-one transformation between elements n of $\mathbb{Z}_m$ (where $m = m_1 \cdot m_2 \cdot \ldots \cdot m_k$) and lists $(r_1, r_2, \ldots, r_k)$ of residues. We shall refer to the list $(r_1, r_2, \ldots, r_k)$ as the *CRT-transform* of n.

The following results are only a few illustrations of the often surprising utility of the Chinese Remainder Theorem.

CRT characterization of Multiplication in $\langle \mathbb{Z}_m, \odot \rangle$: Suppose that the integers $m_1, m_2, \ldots, m_k$ are pairwise-relatively-prime moduli, that $m = m_1 \cdot m_2 \cdot \ldots \cdot m_k$, and that n and $\tilde{n}$ are elements of $\mathbb{Z}_m$ with CRT-transforms $(r_1, r_2, \ldots, r_k)$ and $(\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_k)$, respectively. Then the CRT-transform of the product $n \odot \tilde{n}$ in $\langle \mathbb{Z}_m, \odot \rangle$ is $(r_1 \odot_1 \tilde{r}_1, r_2 \odot_2 \tilde{r}_2, \ldots, r_k \odot_k \tilde{r}_k)$, the componentwise product of the CRT-transforms of n and $\tilde{n}$, where the product $r_i \odot_i \tilde{r}_i$ is a product in $\langle \mathbb{Z}_{m_i}, \odot \rangle$.

Proof:

$$\begin{aligned}
R_{m_i}(n \odot \tilde{n}) &= R_{m_i}(R_m(n \cdot \tilde{n})) \\
&= R_{m_i}(n \cdot \tilde{n} - q \cdot m) \qquad \text{(for some integer } q\text{)} \\
&= R_{m_i}(n \cdot \tilde{n}) \\
&= R_{m_i}(R_{m_i}(n) \cdot R_{m_i}(\tilde{n})) \\
&= R_{m_i}(r_i \cdot \tilde{r}_i) \\
&= r_i \odot_i \tilde{r}_i
\end{aligned}$$

where the first equality follows from the definition of multiplication in $\langle \mathbb{Z}_m, \odot \rangle$, the second from Euclid's division theorem for $\mathbb{Z}$, the third from the fundamental property of remainders and the fact that m is a

multiple of m_i, the fourth from the algebraic property (7.6) of remainders, the fifth from the definitions of the residues r_i and $\tilde{r}_i$, and the sixth from the definition of multiplication in $\langle \mathbb{Z}_{m_i}, \odot \rangle$. □

CRT Characterization of Invertible Elements of $\langle \mathbb{Z}_m, \odot \rangle$: Suppose that $m_1, m_2, \ldots, m_k$ are pairwise relatively prime moduli, that n is an element of $\mathbb{Z}_m$ where $m = m_1 \cdot m_2 \cdot \ldots \cdot m_k$, and that $(r_1, r_2, \ldots, r_k)$ is the CRT-transform of n. Then n is an invertible element of $\langle \mathbb{Z}_m, \odot \rangle$ if and only if r_i is an invertible element of $\langle \mathbb{Z}_{m_i}, \odot \rangle$ for $i = 1, 2, \ldots, k$.

Proof: We note first that the CRT-transform of 1 is just $(1, 1, \ldots, 1)$. Thus, by the CRT characterization of multiplication in $\mathbb{Z}_m$, n is invertible in $\langle \mathbb{Z}_m, \odot \rangle$ if and only if there is a CRT-transform $(\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_k)$ such that $r_i \odot_i \tilde{r}_i = 1$ in $\langle \mathbb{Z}_{m_i}, \odot \rangle$, for $i = 1, 2, \ldots, k$. But such an $(\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_k)$ exists if and only if r_i is invertible in $\langle \mathbb{Z}_{m_i}, \odot \rangle$ for $i = 1, 2, \ldots, k$. □

We can now easily evaluate Euler's function in the general case.

Calculation of Euler's Function: Suppose that $p_1, p_2, \ldots, p_k$ are the distinct prime factors of an integer m, $m \geq 2$. Then

$$\varphi(m) = \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_k}\right) \cdot m. \tag{7.36}$$

Proof: Let $m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$ be the prime factorization of m and define $m_i = p_i^{e_i}$ for $i = 1, 2, \ldots, k$. Then the moduli $m_1, m_2, \ldots, m_k$ are pairwise relatively prime. The CRT-transform $(r_1, r_2, \ldots, r_k)$ corresponds to an invertible element of $\langle \mathbb{Z}_m, \odot \rangle$ if and only if each r_i is an invertible element of $\langle \mathbb{Z}_{m_i}, \odot \rangle$. But, by 7.31, it follows that there are exactly

$$\left(1 - \frac{1}{p_1}\right) \cdot p_1^{e_1} \cdot \left(1 - \frac{1}{p_2}\right) \cdot p_2^{e_2} \cdot \ldots \cdot \left(1 - \frac{1}{p_k}\right) \cdot p_k^{e_k} = \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_k}\right) \cdot m$$

invertible elements of $\langle \mathbb{Z}_m, \odot \rangle$, and this number is $\varphi(m)$. □

7.7 Groups

The true richness of algebra first becomes apparent with the introduction of the next algebraic system, the group.

A *group* is a monoid $\langle G, * \rangle$ such that

(A4) (existence of inverses) every element of G is invertible.

The *order* of a group $\langle G, * \rangle$ is the number of elements in G, $\#(G)$. A *finite group* is simply a group whose order is finite.

Example 7.7.1 The monoid $\langle \mathbb{Z}_m, \odot \rangle$ is *not* a group for any m because the element 0 in $\mathbb{Z}_m$ is not invertible.

Example 7.7.2 Let G be the set of all 2×2 *nonsingular* matrices with entries in the real numbers $\mathbb{R}$ and let $*$ denote matrix multiplication. Then $\langle G, * \rangle$ is an infinite group, i.e., a group of infinite order. That axiom (A1) (closure) holds in $\langle G, * \rangle$ is a consequence of the fact that the determinant of the product of two square matrices is always the product of their determinants; hence the product of two nonsingular

matrices (i.e., matrices with non-zero determinants) is another nonsingular matrix. The reader is invited to check for himself that axioms (A2), (A3) and (A4) also hold in $\langle G, * \rangle$.

An *abelian group (or commutative group)* is a group $\langle G, * \rangle$ such that (A5) (commutative law) for every a and b in G, $a * b = b * a$.

Example 7.7.3 The group $\langle G, * \rangle$ of example 7.7.2 is *not* abelian since the multiplication of 2×2 matrices is not commutative. For instance, choosing

$$a = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } b = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

we find that

$$a * b = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \neq b * a = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

We now define *addition* for the set $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ by the rule

$$a \oplus b = R_m(a + b) \tag{7.37}$$

where the addition on the right is ordinary addition of integers. We will call the operation $\oplus$ *addition modulo* m.

The algebraic system $\langle \mathbb{Z}_m, \oplus \rangle$ is an abelian group of order m whose neutral element is $e = 0$. The inverse of the element a in this group, denoted $\ominus a$, is given by

$$\ominus a = \begin{cases} 0 & \text{if } a = 0 \\ m - a & \text{if } a \neq 0. \end{cases} \tag{7.38}$$

Proof: The proof that $\langle \mathbb{Z}_m, \oplus \rangle$ is a monoid with neutral element $e = 0$ is entirely analogous to the proof, given in section 7.5, that $\langle \mathbb{Z}_m, \odot \rangle$ is a monoid with neutral element 1. From the definition (7.37) of $\oplus$, it is easy to check that (7.38) indeed gives the inverse of any element a in $\mathbb{Z}_m$. Finally, (A5) holds because $a \oplus b = R_m(a + b) = R_m(b + a) = b \oplus a$, where the second equality holds because addition of integers is commutative. □

This is perhaps the place to mention some conventions that are used in algebra. If the group operation is called *addition* (and denoted by some symbol such as $+$ or $\oplus$), then the neutral element is called *zero* (and denoted by some symbol such as 0 or $\mathbf{0}$); the inverse of an element b is called *minus* b (and denoted by some symbol such as $-b$ or $\ominus b$). *Subtraction* is *not* a new operation, rather $a - b$ (or $a \ominus b$) is just a shorthand way to write $a + (-b)$ [or $a \oplus (\ominus b)$]. The group operation is never called addition unless it is commutative, i.e., unless axiom (A5) holds. If the group operation is called *multiplication* (and denoted by some symbol such as $\cdot$ or $\odot$ or $\times$), then the neutral element is called *one*, or the *identity* element (and denoted by some symbol such as 1 or I). The inverse of an element b is called the *reciprocal* of b (or simply *"b inverse"*) and is denoted by some symbol such as $1/b$ or b^{-1}. *Division* in the group is not a new operation; rather a/b is just a shorthand way to write $a \cdot (b^{-1})$ [or $a \cdot (1/b)$ or $a \odot (b^{-1})$, etc.]. The group operation need not be commutative when it is called multiplication, as Examples 7.7.2 and 7.7.3 illustrate.

We have seen that $\langle \mathbb{Z}_m, \odot \rangle$ is not a group. However, the following result tells us how we can "extract" a group from this monoid.

The Group of Invertible Elements of a Monoid: If $\langle M, * \rangle$ is a monoid and M^* is the subset of M consisting of all the invertible elements of the monoid, then $\langle M^*, * \rangle$ is a group.

Proof: The neutral element e of the monoid is always invertible (and is its own inverse) so that M^* is never empty. If a and b are any two elements of M^* with inverses a' and b', respectively, then

$$\begin{aligned}(a * b) * (b' * a') &= a * (b * (b' * a')) \\ &= a * ((b * b') * a') \\ &= a * (e * a') \\ &= a * a' \\ &= e,\end{aligned}$$

where the first two equalities hold because of the associative law (A2), the third and fifth because b and b' are inverse elements as are a and a', and the fourth because e is the neutral element. Similarly, one finds $(b' * a') * (a * b) = e$. Thus, $a * b$ is indeed invertible [and its inverse is $b' * a'$] so that axiom (A1) (closure) holds in $\langle M^*, * \rangle$. The reader is invited to check that axioms (A2), (A3) and (A4) hold also in $\langle M^*, * \rangle$. □

The following is an important special case of the above general result.

The Group of Invertible Elements of $\langle \mathbb{Z}_m, \odot \rangle$: The algebraic system $\langle \mathbb{Z}_m^*, \odot \rangle$, where $\mathbb{Z}_m^*$ is the set of invertible elements of the monoid $\langle \mathbb{Z}_m, \odot \rangle$, is an abelian group of order $\varphi(m)$.

Proof: We showed in the previous section that there are $\varphi(m)$ invertible elements in $\mathbb{Z}_m$, i.e., that $\#(\mathbb{Z}_m^*) = \varphi(m)$. That the group $\langle \mathbb{Z}_m^*, \odot \rangle$ is abelian follows from the fact that $a \odot b = R_m(a \cdot b) = R_m(b \cdot a) = b \odot a$. □

What makes the group structure such a powerful one is the following, almost self-evident, property.

Unique Solvability in Groups: If $\langle G, * \rangle$ is a group and a and b are elements of G, then the equation $a * x = b$ (as well as the equation $x * a = b$) always has a unique solution in G for the unknown x.

Proof: Because $a * (a' * b) = (a * a') * b = e * b = b$, we see that $a' * b$ is indeed a solution in G for x. Conversely, if c is any solution for x in G, then $a * c = b$ so that $a' * (a * c) = a' * b$ and hence $(a' * a) * c = e * c = c = a' * b$; thus, $a' * b$ is the only solution for x in G. □

It is important to note that unique solvability does *not* hold in general in a monoid. For instance, in the monoid $\langle \mathbb{Z}_m, \odot \rangle$, the equation $0 \odot x = 0$ has all m elements of $\mathbb{Z}_m$ as a solution, whereas the equation $0 \odot x = 1$ has no solutions whatsoever in $\mathbb{Z}_m$. Note that it is the unique solvability in a group $\langle G, * \rangle$ that allows one to deduce from $a * b = e$ that $b = a'$ (since $a * x = e$ has the unique solution a') and that allows one to deduce from $a * b = a$ that $b = e$ (since $a * x = a$ has the unique solution e).

The *order of an element* a of a group $\langle G, * \rangle$, denoted $ord(a)$, is the least positive integer n such that

$$a * a * a * \cdots * a = e$$

where there are n occurrences of a on the left – provided such positive integers n exist; *otherwise, the order of a is undefined.* Note that $ord(e) = 1$ so that every group has at least one element of finite order. Note also that the neutral element is the only element of order 1.

Example 7.7.4 Let G be the set of all 2×2 nonsingular *diagonal* matrices with entries in the real numbers $\mathbb{R}$ and let $*$ denote matrix multiplication. Then $\langle G, * \rangle$ is a group whose neutral element is the

identity matrix

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

which has order 1. The matrix $-I$ has order 2 since $(-I) * (-I) = I$. The matrix

$$J = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

also has order 2 since $J * J = I$, as also does the matrix $-J$. The order of all other matrices in this infinite group is undefined. It is interesting to note that the equation $x^2 = I$ has four solutions for the unknown x in the group G, namely the matrices $I, -I, J$ and $-J$. In other words, the matrix I has exactly four square roots.

Hereafter in our discussion of the order of elements of a group, we will think of the group operation $*$ as a kind of "multiplication" and hence will write $a * a = a^2, a * a * a = a^3$, etc. . We will similarly write the inverse of a as a^{-1} and also write $a^{-1} * a^{-1} = a^{-2}, a^{-1} * a^{-1} * a^{-1} = a^{-3}$, etc. . This is justified since, for instance,

$$\begin{aligned} a^2 * a^{-2} &= (a * a) * (a^{-1} * a^{-1}) \\ &= a * (a * a^{-1}) * a^{-1} \\ &= a * a^{-1} \\ &= e. \end{aligned}$$

We see that it is also consistent to write the neutral element e as a^0. The reader is asked to remember, however, that we are actually considering arbitrary groups so that in fact $*$ might be some form of "addition".

It is easy to see that *every element of a finite group has a well-defined order.* For suppose that a is in a group $\langle G, * \rangle$ of order n. Then, for some integers i and j with $0 \leq i < j \leq n$, it must hold that $a^i = a^j$ and hence that $a^{j-i} = e$.

By the convention just described, if a is any element of the arbitrary "multiplicative" group $\langle G, * \rangle$, then a^i is well defined for every i in $\mathbb{Z}$. Moreover, the usual rules of exponents hold, i.e.,

$$a^i * a^j = a^{i+j} \tag{7.39}$$

and

$$(a^i)^j = a^{i \cdot j}. \tag{7.40}$$

Fundamental Property of the Order of a Group Element: If a has order n in the "multiplicative" group $\langle G, * \rangle$, then the n elements $a^0 = e, a^1 = a, a^2, \ldots, a^{n-1}$ are all distinct. Moreover, for every i in $\mathbb{Z}$,

$$a^i = a^{R_n(i)}. \tag{7.41}$$

Proof: Suppose that $a^i = a^j$ where $0 \leq i \leq j < n$. Then, multiplying both sides of this equation by a^{-i} gives $e = a^{j-i}$ where $0 \leq j - i < n$. By the definition of $\text{ord}(a)$, it follows that $j = i$, and hence that $a^0 = e, a^1 = a, a^2, \ldots, a^{n-1}$ are all distinct as claimed. Euclid's division theorem states that every integer i can be written as $i = q \cdot n + r$, where $r = R_n(i)$. Thus, $a^i = a^{q \cdot n + r} = a^{q \cdot n} * a^r = (a^n)^q * a^r =$

$e^q * a^r = e * a^r = a^r$ where the second equality follows from (7.39), the third from (7.40), the fourth from the fact that $n = \text{ord}(a)$, and the remaining equalities are obvious. But $r = R_n(i)$, so we have proved (7.41). □

The simplest, but one of the most interesting, groups is the *cyclic group*, which is defined as follows. The *cyclic group of order* n is a "multiplicative" group $\langle G, * \rangle$ that contains an element of order n so that $G = \{a^0 = e, a, a^2, \ldots, a^{n-1}\}$. Any element of order n in G is said to be a *generator* of the cyclic group. Because

$$a^i * a^j = a^{i+j} = a^{R_n(i+j)} = a^{i \oplus j}$$

where $\oplus$ denotes addition modulo n [and where the first equality follows from (7.39), the second from (7.41), and the third from the definition (7.37)], it follows that the cyclic group of order n is essentially the same as the group $\langle \mathbb{Z}_n, \oplus \rangle$, which is why one speaks of *the* cyclic group of order n. Note that *the cyclic group is always an abelian group.* The main features of the cyclic group are summarized in the following statement.

Properties of the Cyclic Group: If a is a generator of the cyclic group of order n, then the element $b = a^i$, where $0 \leq i < n$, has order $n/\gcd(n, i)$. In particular, b is also a generator of the cyclic group of order n if and only if $\gcd(n, i) = 1$ so that the cyclic group of order n contains exactly $\varphi(n)$ generators.

Proof: Because $b^k = a^{ik}$ and because $a^{ik} = e$ if and only if n divides $i \cdot k$, it follows that the order k of b is the smallest positive integer k such that $i \cdot k$ is divisible by n (as well of course as by i), i.e., $i \cdot k = \text{lcm}(n, i)$. But then $k = \text{lcm}(n, i)/i = n/\gcd(n, i)$, as was to be shown. □

Example 7.7.5 Consider the group $\langle \mathbb{Z}_m^*, \odot \rangle$ of invertible elements of the monoid $\langle \mathbb{Z}_m, \odot \rangle$. Because 7 is a prime, $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$. Note that $3^1 = 3, 3^2 = 2, 3^3 = 6, 3^4 = 4, 3^5 = 5, 3^6 = 1$. Hence $\langle \mathbb{Z}_7^*, \odot \rangle$ is the cyclic group of order 6 and 3 is a generator of this cyclic group. This cyclic group has $\varphi(6) = \varphi(3 \cdot 2) = 2 \cdot 1 = 2$ generators. The other generator is $3^5 = 5$. because $\gcd(6, 2) = \gcd(6, 4) = 2$, the elements $3^2 = 2$ and $3^4 = 4$ both have order $6/2 = 3$. Because $\gcd(6, 3) = 3$, the element $3^3 = 6$ has order $6/3 = 2$. The element $3^0 = 1$ has of course order 1.

Uniqueness of the Cyclic Group of Order *n*: The cyclic group of order n is essentially the group $\langle \mathbb{Z}_n, \oplus \rangle$, i.e., if $\langle G, * \rangle$ is the cyclic group of order n then there is an invertible function $f : G \to \mathbb{Z}_n$ such that $f(\alpha * \beta) = f(\alpha) \oplus f(\beta)$.

7.8 Subgroups, Cosets and the Theorems of Lagrange, Fermat and Euler

If $\langle G, * \rangle$ is a group and H is a subset of G such that $\langle H, * \rangle$ is also a group, then $\langle H, * \rangle$ is called a *subgroup* of the group $\langle G, * \rangle$. The subsets $H = \{e\}$ and $H = G$ always and trivially give subgroups, and these two trivial subgroups are always different unless $G = \{e\}$. The following simple, but important result, should be self-evident and its proof is left to the reader.

Subgroups Generated by Elements of Finite Order: If $\langle G, * \rangle$ is a group and a is an element of G with order n, then $\langle H, * \rangle$, where $H = \{a^0 = e, a, a^2, \ldots, a^{n-1}\}$, is a subgroup of $\langle G, * \rangle$; moreover, this subgroup is the cyclic group of order n generated by a, and hence is abelian regardless of whether or not $\langle G, * \rangle$ is abelian.

If $\langle G, * \rangle$ is a group, S is any subset of G, and g any element of G, then one writes $g * S$ to mean the set $\{g * s : s \text{ in } S\}$ and $S * g$ is similarly defined. Note that the associative law (A2) implies that

$$g_1 * (g_2 * S) = (g_1 * g_2) * S \tag{7.42}$$

holds for all g_1 and g_2 in G.

If $\langle H, * \rangle$ is a subgroup of $\langle G, * \rangle$ and g is any element of G, then the set $g * H$ is called a *right coset of the group* $\langle G, * \rangle$ *relative to the subgroup* $\langle H, * \rangle$. Similarly, $H * g$ is called a *left coset of the group* $\langle G, * \rangle$ *relative to the subgroup* $\langle H, * \rangle$. If $\langle G, * \rangle$ is abelian, then of course $g * H = H * g$, but in general $g * H \neq H * g$. However, if h is an element of H, then

$$h * H = H * h = H, \tag{7.43}$$

as follows easily from unique solvability in groups. Hereafter, for convenience, we will consider only right cosets, but it should be clear that all results stated apply also to left cosets *mutatis mutandis*.

Fundamental Properties of Cosets: If $\langle H, * \rangle$ is a subgroup of $\langle G, * \rangle$ and g is any element of G, then

$$\#(g * H) = \#(H),$$

i.e., all right cosets have the same cardinality as the subgroup to which they are relative. Moreover, if g_1 and g_2 are any elements of G, then

$$\text{either} \quad g_1 * H = g_2 * H \quad \text{or} \quad (g_1 * H) \cap (g_2 * H) = \varnothing,$$

i.e., two right cosets are either identical or disjoint.

Proof: The mapping $f(h) = g * h$ is an invertible mapping (or *bijection*) from H to the coset $g * H$, as follows by unique solvability in groups. Thus, $\#(g*H) = \#(H)$. Suppose next that $(g_1*H) \cap (g_2*H) \neq \varnothing$, i.e., that these two right cosets are not disjoint. Then there exist h_1 and h_2 in H such that $g_1 * h_1 = g_2 * h_2$. But then $g_2 = g_1 * h_3$, where $h_3 = h_1 * h_2^{-1}$ and hence $g_2 * H = (g_1 * h_3) * H = g_1 * (h_3 * H) = g_1 * H$ [where the second equality holds because of (7.42) and the third because of (7.43)]. Thus these two right cosets are identical. □

If $\langle H, * \rangle$ is a subgroup of $\langle G, * \rangle$, then the neutral element e of G must also be in H, and hence the element g of G must lie in the coset $g * H$. Thus, one can speak unambiguously of *the right coset that contains* g.

We are now in position to prove one of the most beautiful and most important results in algebra.

Lagrange's Theorem: If $\langle H, * \rangle$ is a subgroup of the *finite* group $\langle G, * \rangle$, then $\#(H)$ divides $\#(G)$, i.e., *the order of a subgroup of a finite group always divides the order of the group.*

Proof: Choose $g_1 = e$ and let $C_1 = g_1 * H = H$. If $G \setminus C_1$ is not empty (where $\setminus$ denotes set subtraction, i.e., $A \setminus B$ is the set of those elements of A that are not also elements of B), choose g_2 as any element of $G \setminus C_1$ and let $C_2 = g_2 * H$. Continue in this manner by choosing g_{i+1} to be any element of $G \setminus (C_1 \cup C_2 \cup \cdots \cup C_i)$ as long as this set is nonempty, and letting $C_{i+1} = g_{i+1} * H$. When this procedure terminates [which it must since G is finite], say after g_q is chosen, the cosets $C_1, C_2, \ldots, C_q$ thus obtained satisfy $G = C_1 \cup C_2 \cup \cdots \cup C_q$. But, by the fundamental properties of cosets, these q cosets are pairwise disjoint and each contains exactly $\#(H)$ elements. Thus, it must be true that $\#(G) = q \cdot \#(H)$, i.e., that $\#(H)$ divides $\#(G)$. □

Corollary to Lagrange's Theorem: The order of every element of a finite group divides the order of the group.

Proof: If a has order n in the group $\langle G, * \rangle$, then a generates the cyclic group of order n, which is a subgroup of $\langle G, * \rangle$. Thus n divides $\#(G)$ by Lagrange's theorem. □

Two special cases of this corollary are of great interest in cryptography.

Euler's Theorem: If b is any invertible element in $\langle \mathbb{Z}_m, \odot \rangle$, then $b^{\varphi(m)} = 1$.

Proof: By hypothesis, b is an element in the abelian group $\langle \mathbb{Z}_m, \odot \rangle$, which has order $\varphi(m)$. Hence the order n of b divides $\varphi(m)$, i.e., $\varphi(m) = n \cdot q$. Thus,

$$b^{\varphi(m)} = b^{n \cdot q} = (b^n)^q = 1^q = 1.$$

□

As we will later see, Euler's Theorem is the foundation for the Rivest-Shamir-Adleman (RSA) public-key cryptosystem, which is widely regarded as the best of the presently known public-key cryptosystems.

Fermat's Theorem: If p is a prime and b is any non-zero element of $\mathbb{Z}_p$, then $b^{(p-1)} = 1$.

Proof: This is just a special case of Euler's Theorem since, for a prime p, $\varphi(p) = p - 1$ and every non-zero element of $\mathbb{Z}_p$ is invertible. □

Fermat's Theorem is the foundation for *primality testing*. Suppose that m is any modulus and b is any non-zero element of $\mathbb{Z}_m$. If we calculate b^{m-1} in $\langle \mathbb{Z}_m, \odot \rangle$ and find that the result is *not* 1, then Fermat's Theorem assures us that m is *not* a prime. Such reasoning will prove to be very useful when we seek the large (about 100 decimal digits) randomly-chosen primes required for the RSA public-key cryptosystem.

We have seen that Fermat's Theorem is a special case of Euler's Theorem, which in turn is a special case of Lagrange's Theorem. Why Fermat and Euler are nonetheless given credit will be apparent to the reader when one considers the life spans of Fermat (1601-1665), Euler (1707-1783) and Lagrange (1736-1813).

7.9 Rings

We now consider an algebraic system with two operations.

A *ring* is an algebraic system $\langle R, \oplus, \odot \rangle$ such that

(i) $\langle R, \oplus \rangle$ is an abelian group, whose neutral element is denoted 0;

(ii) $\langle R, \odot \rangle$ is a monoid whose neutral element, which is denoted by 1, is different from 0; and

(iii) (distributive laws) for every a, b and c in R,

$$a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c) \tag{7.44}$$

$$(b \oplus c) \odot a = (b \odot a) \oplus (c \odot a). \tag{7.45}$$

A *commutative ring* is a ring $\langle R, \oplus, \odot \rangle$ such that $\odot$ is commutative, i.e., such that, for every a and b in R, $a \odot b = b \odot a$. For a commutative ring, of course, only one of the two distributive laws is required, since either one then implies the other.

The *units* of a ring are the invertible elements of the monoid $\langle R, \odot \rangle$. The set of units is denoted R^* and we recall that $\langle R^*, \odot \rangle$ is a group.

Example 7.9.1 $\langle \mathbb{Z}, +, \cdot \rangle$, where $+$ and $\cdot$ denote ordinary integer addition and multiplicaton, respectively, is a commutative ring whose units are the elements 1 and -1, i.e., $\mathbb{Z}^* = \{-1, 1\}$. This ring is called simply the *ring of integers*.

Example 7.9.2 $\langle \mathbb{Z}_m, \oplus, \odot \rangle$, where $\oplus$ and $\odot$ denote addition modulo m and multiplication modulo m, respectively, is a commutative ring and is called *the ring of integers modulo m*. The set of units is $\mathbb{Z}_m^* = \{u : u \text{ is in } \mathbb{Z}_m \text{ and } \gcd(m, u) = 1\}$.

We have already proved all the claims of this example except to prove that part (iii) (the distributive laws) of the definition of a ring is satisfied. To prove this, suppose that a, b and c are in $\mathbb{Z}_m$, then

$$\begin{aligned} a \odot (b \oplus c) &= R_m(R_m(a) \cdot R_m(b \oplus c)) \\ &= R_m(R_m(a) \cdot R_m(b + c)) \\ &= R_m(a \cdot (b + c)) \end{aligned}$$

where the first and third equalities follow from the algebraic property (7.6) of remainders and the second from the definition (7.37) of $\oplus$. Similarly,

$$\begin{aligned} (a \odot b) \oplus (a \odot c) &= R_m(R_m(a \odot b) + R_m(a \odot c)) \\ &= R_m(R_m(a \cdot b) + R_m(a \cdot c)) \\ &= R_m(a \cdot b + a \cdot c) \\ &= R_m(a \cdot (b + c)) \end{aligned}$$

where the first and third equalities follow from the algebraic property (7.5) of remainders, the second from the definition of $\odot$, and the last from the fact that the distributive law holds in $\langle \mathbb{Z}, +, \cdot \rangle$. Thus $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$. Because $\odot$ is commutative, there is no need to prove the other distributive law. □

The ring of integers modulo m is by far the most important algebraic system used in contemporary cryptography!

Many "standard" rules of algebraic manipulation are valid in all rings.

Multiplication by 0 and Rules of Signs: If $\langle R, \oplus, \odot \rangle$ is a ring and if a and b are elements of R, then

$$a \odot 0 = 0 \odot a = 0 \tag{7.46}$$

$$\ominus(a \odot b) = (\ominus a) \odot b = a \odot (\ominus b) \tag{7.47}$$

$$(\ominus a) \odot (\ominus b) = a \odot b. \tag{7.48}$$

Proof: Because $0 = 0 \oplus 0$, the distributive law (7.44) gives $a \odot 0 = (a \odot 0) \oplus (a \odot 0)$. Unique solvability in the group $\langle R, \oplus \rangle$ thus implies $a \odot 0 = 0$, the proof that $0 \odot a = 0$ is similar. Next, we note that

$((\ominus a) \odot b) \oplus (a \odot b) = (\ominus a \oplus a) \odot b = 0 \odot b = 0$ where the first equality follows from the distributive law (7.44). By unique solvability in the group $\langle R, \oplus \rangle$, it follows that $(\ominus a) \odot b = \ominus(a \odot b)$. The proof that $a \odot (\ominus b) = \ominus(a \odot b)$ is similar. Finally, we note that (7.47) implies

$$(\ominus a) \odot (\ominus b) = \ominus(a \odot (\ominus b)) = \ominus(\ominus(a \odot b)) = a \odot b.$$

□

7.10 Fields

A *field* is an algebraic system $\langle F, +, \cdot \rangle$ such that $\langle F, +, \cdot \rangle$ is a commutative ring and every non-zero element of the ring is a unit.

An alternative, but fully equivalent, definition (as the reader should have no difficulty verifying) of a field is the following one.

A field is an algebraic system $\langle F, +, \cdot \rangle$ such that

(i) $\langle F, + \rangle$ is an abelian group (whose neutral element is denoted by 0);

(ii) $\langle F \setminus \{0\}, \cdot \rangle$ is an abelian group (whose neutral element is denoted by 1); and

(iii) for every a, b and c in F, the distributive law

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c) \tag{7.49}$$

holds, and multiplication by 0 obeys the rule

$$a \cdot 0 = 0 \cdot a = 0. \tag{7.50}$$

In this alternative definition of a field, it is necessary to define multiplication by 0 with the axiom (7.50) because multiplication by 0 is not defined in part (ii). The first definition of a field does not require this, as the fact that $\langle F, \cdot \rangle$ is a monoid means that multiplication by 0 is well defined.

The reader is certainly already acquainted with several fields, namely

(1) *the field* $\langle \mathbb{Q}, +, \cdot \rangle$ *of rational numbers;*

(2) *the field* $\langle \mathbb{R}, +, \cdot \rangle$ *of real numbers; and*

(3) *the field* $\langle \mathbb{C}, +, \cdot \rangle$ *of complex numbers.*

If $\langle E, +, \cdot \rangle$ is a field and if F is a subset of E such that $\langle F, +, \cdot \rangle$ is a field, then $\langle F, +, \cdot \rangle$ is called a *subfield* of $\langle E, +, \cdot \rangle$, and $\langle E, +, \cdot \rangle$ is called an *extension field* of $\langle F, +, \cdot \rangle$. We see that the rational field is a subfield of the real field and that the complex field is an extension field of the real field. Of course the rational field is also a subfield of the complex field, and the complex field is also an extension field of the rational field.

The reason that we have used the "standard" signs for addition and multiplication in a field is to stress the fact that *all the standard rules* (the associative, commutative and distributive laws) *apply to*

addition, subtraction, multiplication and division (by non-zero numbers). Thus, sets of linear equations are solved the same way in any field; in particular, *determinants and matrices are defined in the same way and have the same properties* (e.g., the determinant of the product of square matrices is the product of the determinants) *in any field.* We assume that the reader is familiar with the elementary properties of such determinants and matrices.

A field $\langle F, +, \cdot \rangle$ for which F is a finite set is called a *finite field* or a *Galois field.* The finite fields are by far the most important algebraic systems used in contemporary coding theory! We already know how to construct some finite fields.

Prime Field Theorem: The commutative ring $\langle \mathbb{Z}_m, \oplus, \odot \rangle$ is a field if and only if m is a prime.

Proof: We saw in Section 7.5 that every non-zero element of the monoid $\langle \mathbb{Z}_m, \odot \rangle$ is invertible if and only if m is a prime. □

We will denote the finite field $\langle \mathbb{Z}_p, \oplus, \odot \rangle$ where p is a prime by the symbol $GF(p)$ to give due honor to their discoverer, Evariste Galois (1811-1832). Galois, in his unfortunately short lifetime, managed to find all the finite fields. We shall also write $\oplus$ and $\odot$ simply by $+$ and $\cdot$ (or simply by juxtaposition of elements), respectively, in $GF(p)$. The modern trend seems to be to denote $GF(p)$ as $\mathbb{F}_p$, but this is a trend that admirers of Galois (such as this writer) believe should be resisted.

The *characteristic* of a field is the order of 1 in the additive group $\langle F, + \rangle$ of the field provided that this order is defined, and is said to be 0 [beware, some writers say ∞] otherwise. The characteristic of $GF(p)$ is p. The characteristic of the rational, real and complex fields is 0 [or, if you prefer, ∞].

The characteristic of a field is either 0 [or, if you prefer, ∞] *or a prime p.*

Proof: Suppose the order of 1 in the additive group $\langle F, + \rangle$ is a composite integer m, i.e., $m = m_1 \cdot m_2$ where $m_1 \geq 2$ and $m_2 \geq 2$. Writing i to denote the sum of i 1's in the field $\langle F, +, \cdot \rangle$, we see that m_1 and m_2 are non-zero elements of the field, but $m_1 \cdot m_2 = 0$, contradicting the fact that $\langle F \setminus \{0\}, \cdot \rangle$ is a group. □

We will see later that there exists a finite field with q elements if and only if q is a power of a prime, i.e., if and only if $q = p^m$ where p is a prime and m is a positive integer, and that this field is essentially unique. We will denote this field by $GF(q)$ or $GF(p^m)$. (We will also see that there are infinite fields whose characteristic is a prime p.)

7.11 Vector Spaces

We now consider an algebraic system with two sets of elements and four operations.

A *vector space* is an algebraic system $\langle \mathbf{V}, +, \cdot, F, +, \cdot \rangle$ such that

(i) $\langle F, +, \cdot \rangle$ is a field;

(ii) $\langle \mathbf{V}, + \rangle$ is an abelian group; (whose neutral element we denote by $\mathbf{0}$); and

(iii) $\cdot$ is an operation on pairs consisting of an element of F and an element of $\mathbf{V}$ such that, for all c_1, c_2

in F and all $\mathbf{v}_1, \mathbf{v}_2$ in $\mathbf{V}$,

(1)	(closure)	$c_1 \cdot \mathbf{v}_1$ is in $\mathbf{V}$,	
(2)	(scaling by 1)	$1 \cdot \mathbf{v}_1 = \mathbf{v}_1$,	
(3)	(associativity)	$c_1 \cdot (c_2 \cdot \mathbf{v}_1) = (c_1 \cdot c_2) \cdot \mathbf{v}_1$, and	
(4)	(distributivity)	$(c_1 + c_2) \cdot \mathbf{v}_1 = (c_1 \cdot \mathbf{v}_1) + (c_2 \cdot \mathbf{v}_1)$	(7.51)
		$c_1 \cdot (\mathbf{v}_1 + \mathbf{v}_2) = (c_1 \cdot \mathbf{v}_1) + (c_1 \cdot \mathbf{v}_2)$.	(7.52)

It should be noticed that the "associativity" defined in (iii)(3) is essentially different from the axiom (A2) of the associative law in that, for the former, the first operation on the left is different from the first operation on the right. Similarly, the addition operation on the left in the "distributivity" equation (7.51) is different from the addition operation on the right.

In a vector space $\langle \mathbf{V}, +, \cdot, F, +, \cdot \rangle$, the elements of $\mathbf{V}$ are called *vectors* and the elements of F are called *scalars*. Note that axiom (iii)(1) specifies that $\mathbf{V}$ must be closed under multiplication of its elements by scalars. Axiom (iii)(2) may seem redundant to the reader, but in fact one can construct algebraic systems that do not satisfy (iii)(2) but do satisfy all the other axioms of a vector space.

We now show that certain "natural" rules hold in all vector spaces.

Scaling by 0 and by the Negatives of Scalars: If c is any scalar and $\mathbf{v}$ is any vector in a vector space $\langle \mathbf{V}, +, \cdot, F, +, \cdot \rangle$, then

$$0 \cdot \mathbf{v} = \mathbf{0} \tag{7.53}$$

and

$$(-c) \cdot \mathbf{v} = -(c \cdot \mathbf{v}). \tag{7.54}$$

Proof: We note first that $(0 + 0) \cdot \mathbf{v} = 0 \cdot \mathbf{v}$. By the distributivity relation (7.51), it follows that $(0 \cdot \mathbf{v}) + (0 \cdot \mathbf{v}) = 0 \cdot \mathbf{v}$. Unique solvability in the group $\langle \mathbf{V}, + \rangle$ now implies that $0 \cdot \mathbf{v} = \mathbf{0}$.

We note next that $c \cdot \mathbf{v} + (-c) \cdot \mathbf{v} = (c - c) \cdot v = 0 \cdot \mathbf{v} = \mathbf{0}$ where the first equality follows from (7.51). Unique solvability in the group $\langle \mathbf{V}, + \rangle$ now implies that $-(c \cdot \mathbf{v}) = (-c) \cdot \mathbf{v}$. □

Scaling the Zero Vector: For every scalar c in a vector space $\langle \mathbf{V}, +, \cdot, F, +, \cdot \rangle$

$$c \cdot \mathbf{0} = \mathbf{0}. \tag{7.55}$$

Proof: We note first that $c \cdot (\mathbf{0} + \mathbf{0}) = c \cdot \mathbf{0}$. By the distributivity relation (7.54), it follows that $c \cdot \mathbf{0} + c \cdot \mathbf{0} = c \cdot \mathbf{0}$. Unique solvability in the group $\langle \mathbf{V}, + \rangle$ now implies that $c \cdot \mathbf{0} = \mathbf{0}$. □

Hereafter, we will often write $c \cdot \mathbf{v}$ simply as $c\mathbf{v}$, but the reader should always remember that the implied multiplication in $c\mathbf{v}$ is the multiplication of a vector by a scalar, *not* the multiplication of two scalars. Similarly, we will often write $c_1 \cdot c_2$ simply as $c_1 c_2$, but the reader should always remember that the implied multiplication in $c_1 c_2$ is multiplication of two scalars. From the context, it will always be clear which type of multiplication is understood.

If n is a positive integer and if $c_1, c_2, \ldots, c_n$ are scalars, then the vector

$$c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \cdots + c_n \mathbf{v}_n$$

is called a *linear combination* of the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. Note that if $c_1 = c_2 = \cdots = c_n = 0$, then the above linear combination is the vector $\mathbf{0}$ regardless of the values of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$.

The vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are said to be *linearly independent* if the only linear combination of these vectors that gives $\mathbf{0}$ is that with $c_1 = c_2 = \cdots = c_n = 0$; otherwise they are said to be *linearly dependent*. Note that the zero vector $\mathbf{0}$ is by itself linearly dependent because $1 \cdot \mathbf{0} = \mathbf{0}$; but every non-zero vector $\mathbf{v}$ is by itself linearly independent because, for any non-zero scalar c, $c \cdot \mathbf{v} = \mathbf{0}$ would imply $c^{-1} \cdot (c \cdot \mathbf{v}) = c^{-1} \cdot \mathbf{0}$ and thus $(c^{-1} \cdot c) \cdot \mathbf{v} = 1 \cdot \mathbf{v} = \mathbf{v} = c^{-1} \cdot \mathbf{0} = \mathbf{0}$ contradicting the fact that $\mathbf{v} \neq \mathbf{0}$.

If $\langle \mathbf{V}, +, \cdot, F, +, \cdot \rangle$ is a vector space and if $\mathbf{U}$ is a subset of $\mathbf{V}$ such that $\langle \mathbf{U}, +, \cdot, F, +, \cdot \rangle$ is also a vector space, then $\langle \mathbf{U}, +, \cdot, F, +, \cdot \rangle$ is called a *subspace* of the vector space $\langle \mathbf{V}, +, \cdot, F, +, \cdot \rangle$. Hereafter, we will generally follow customary, but imprecise terminology and speak simply of the "vector space $\mathbf{V}$" or the "subspace $\mathbf{U}$" and we will refer to the field $\langle F, +, \cdot \rangle$ as simply the "scalar field F".

Test for a Subspace: A non-empty subset $\mathbf{U}$ of a vector space $\mathbf{V}$ is a subspace if and only if (1) for all $\mathbf{u}_1$ and $\mathbf{u}_2$ in $\mathbf{U}$, $\mathbf{u}_1 + \mathbf{u}_2$ is also in $\mathbf{U}$, and (2) for every c in the scalar field F and every $\mathbf{u}$ in $\mathbf{U}$, $c \cdot \mathbf{u}$ is also in $\mathbf{U}$ [i.e., if and only if $\mathbf{U}$ is closed under vector addition and closed under multiplication by scalars.]

Proof: If $\mathbf{U}$ is a subspace of $\mathbf{V}$, then (1) and (2) hold trivially. Suppose conversely that (1) and (2) hold for a non-empty subset $\mathbf{U}$ of $\mathbf{V}$. Let $\mathbf{u}$ be any vector in $\mathbf{U}$. Then (2) implies that $0\mathbf{u} = \mathbf{0}$ and $(-1)\mathbf{u} = -\mathbf{u}$ are in $\mathbf{U}$. Closure of $+$ holds in $\mathbf{U}$ because of (1), and the associative law holds in $\mathbf{U}$ because it holds in the larger set $\mathbf{V}$. Morever, the operation $+$ is commutative. It follows that axioms (A1) - (A4) hold in $\langle \mathbf{U}, + \rangle$, which is thus an abelian group. Axiom (iii) of a vector space holds for $\mathbf{U}$ because of (2). The remaining parts of Axiom (iii) also hold in $\mathbf{U}$ because they hold in the larger set $\mathbf{V}$. Thus, $\mathbf{U}$ is indeed a vector space. □

It follows from the subspace test that the subset $\mathbf{U} = \{\mathbf{0}\}$ is trivially a subspace of every vector space. If n is a positive integer and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are any vectors in a vector space $\mathbf{V}$, then the set $\{c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n : c_1, c_2, \ldots, c_n$ are in the scalar field $F\}$ of all linear combinations of the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is a subspace of $\mathbf{V}$, as follows easily from the subspace test. This subspace is called the *subspace spanned by* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ and will be denoted $S(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$.

If $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are linearly independent and $\mathbf{V} = S(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$, then $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are called a *basis* for $\mathbf{V}$; moreover, every choice of n linearly independent vectors in $\mathbf{V}$ is then a basis for $\mathbf{V}$. The *dimension* of a vector space $\mathbf{V}$, denoted $\dim(\mathbf{V})$, is the maximum number of linearly independent vectors that can be chosen from $\mathbf{V}$. It follows that $\dim(\{\mathbf{0}\}) = 0$ and that the dimension of a finite-dimensional vector space is equal to the number of vectors n in a basis.

We now consider the vector space of greatest importance in coding theory. For a field F, we write F^N to denote the set of N-*tuples* (or "row vectors") $[b_1, b_2, \ldots, b_N]$ whose components are elements of F. We define addition of N-tuples and multiplication of N-tuples by a scalar (i.e., by an element of F) by the rules

$$[b_1, b_2, \ldots, b_N] + [b'_1, b'_2, \ldots, b'_N] = [b_1 + b'_1, b_2 + b'_2, \ldots, b_N + b'_N] \tag{7.56}$$

and

$$c \cdot [b_1, b_2, \ldots, b_N] = [cb_1, cb_2, \ldots, cb_N], \tag{7.57}$$

respectively.

Vector Space of N-Tuples: For any positive integer N, the set F^N is an N-dimensional vector space with $\mathbf{0} = [0, 0, \ldots, 0]$. The vectors $[1, 0, \ldots, 0], [0, 1, \ldots, 0], \ldots, [0, 0, \ldots, 1]$ containing a single non-zero component equal to 1 are a basis (called the *canonical basis*) for F^N.

Proof: The reader is invited to make the simple check that F^N indeed satisfies the axioms of a vector space and that $\mathbf{0} = [0, 0, \ldots, 0]$. Because $c_1[1, 0, \ldots, 0] + c_2[0, 1, \ldots, 0] + \cdots + c_N[0, 0, \ldots, 1] = [c_1, c_2, \ldots, c_N]$, it

follows that this linear combination is $\mathbf{0} = [0, 0, \ldots, 0]$ if and only if $c_1 = c_2 = \cdots = c_N = 0$ so that these N vectors are linearly independent, and it follows further that these N vectors span F^N. Thus, they are indeed a basis for F^N. □

If $\mathbf{V}$ is any vector space with $0 < n < \infty$ (i.e., any non-trivial finite-dimensional vector space) with the scalar field F and if $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is any basis for $\mathbf{V}$, then *every vector* $\mathbf{v}$ in $\mathbf{V}$ *can be written uniquely as a linear combination of the basis vectors*,

$$\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n \tag{7.58}$$

as follows from the fact that if two such linear combinations were equal then their difference would be a non-trivial linear combination equal to $\mathbf{0}$, which would contradict the linear independence of the vectors in a basis. If we now associate the vector $\mathbf{v}$ in $\mathbf{V}$ with the n-tuple $[c_1, c_2, \ldots, c_n]$, then we have created an invertible mapping (or bijection) between $\mathbf{V}$ and F^n. Moreover, addition in $\mathbf{V}$ and multiplication of vectors in $\mathbf{V}$ by scalars can equivalently be carried out on their "transforms" in F^n. It follows that *every n-dimensional vector space over the scalar field F is essentially the same as the vector space F^n* (the technical term in algebra is that $\mathbf{V}$ is *isomorphic* to F^n). We shall not be interested in isomorphisms of vector spaces; the interesting applications of vector spaces in coding and cryptography rely on the specific basis-dependent properties of the spaces, not on their "abstract equivalence" to F^n.

7.12 Formal Power Series and Polynomials

A *formal power series* over the *field* F in the *indeterminate* X is a "sum" $A(X)$ of the form

$$A(X) = a_0 + a_1X + a_2X^2 + \cdots$$

where the order of the terms in this "sum" is arbitrary and where the coefficient a_i is in F for $0 \le i < \infty$. One often writes simply

$$A(X) = \sum_{i=0}^{\infty} a_iX^i.$$

The equality $A(X) = B(X)$ of two such formal power series means just that the coefficient of X^i in each is the same for all i, i.e., that $a_i = b_i$ for $0 \le i < \infty$. The *sum* $C(X) = A(X) + B(X)$ of two formal power series is defined by $c_i = a_i + b_i$ for $0 \le i < \infty$. The *product* of two formal series is defined by

$$c_i = \sum_{j=0}^{i} a_jb_{i-j}, \qquad 0 \le i < \infty. \tag{7.59}$$

Note that c_i is a sum of finitely many elements of F and hence is a well-defined element of F for every $i, 0 \le i < \infty$.

A *polynomial* over the *field* F in the indeterminate X is a formal power series in which only a finite number (perhaps none) of the coefficients are non-zero. The polynomial with all-zero coefficients is written simply as 0. The *degree* of a polynomial $A(X), A(X) \neq 0$, is the greatest i such that $a_i \neq 0$; the degree of the polynomial 0 is, by way of convention, taken to be $-\infty$. The following property is an immediate consequence of (7.59).

Degree Property of Polynomials: if $A(X)$ and $B(X)$ are polynomials over the field F, then

$$\deg[A(X)B(X)] = \deg[A(X)] + \deg[B(X)]$$

where deg[·] denotes the degree of the enclosed polynomial.

One writes $F[X]$ to denote the set of all polynomials over the field F in the indeterminate X. The following theorem can be proved entirely analogously to Euclid's Division Theorem for the Integers in Section 7.2.

The Division Theorem for F[X]: Given polynomials N(X) (the "dividend") and $D(X)$ (the "divisor") with $D(X) \neq 0$, there exist unique polynomials $Q(X)$ (the "quotient") and $R(X)$ (the "remainder") such that

$$N(X) = Q(X)D(X) + R(X) \tag{7.60}$$

and

$$\deg[R(X)] < \deg[D(X)]. \tag{7.61}$$

We write $R_{D(X)}[N(X)]$ to denote the remainder when $N(X)$ is divided by $D(X)$, assuming that $D(X) \neq 0$ when we use this notation. Remainders in $F[X]$ satisfy much the same properties as remainders in $\mathbb{Z}$, which were discussed in Section 7.2. The reader is invited to make simple proofs of the following properties and theorems.

Fundamental Properties on Remainders in F[X]: For any polynomials $N(X)$ and $I(X)$,

$$R_{D(X)}[N(X) + I(X)D(X)] = R_{D(X)}[N(X)]. \tag{7.62}$$

Algebraic Properties of Remainders in F[X]: For any polynomials $A(X)$ and $B(X)$,

$$R_{D(X)}[A(X) + B(X)] = R_{D(X)}[A(X)] + R_{D(X)}[B(X)] \tag{7.63}$$

and

$$R_{D(X)}[A(X)B(X)] = R_{D(X)}[R_{D(X)}[A(X)]R_{D(X)}[B(X)]\,]. \tag{7.64}$$

Remark: Note that we need not take the remainder of the sum on the right in (7.63) because the degree of this sum is already less than $\deg[D(X)]$ and hence this sum is its own remainder when divided by $D(X)$.

The *leading coefficient* of a non-zero polynomial $A(X)$ is the coefficient of X^d in this polynomial where $d = \deg[A(X)]$. (The polynomial 0 has no leading coefficient.) A polynomial is said to be *monic* if its leading coefficient is 1.

One says that the non-zero polynomial $D(X)$ *divides* the polynomial $N(X)$ just in case that $R_{D(X)}[N(X)] = 0$ or, equivalently, if there is a polynomial $Q(X)$ such that $N(X) = Q(X)D(X)$. Note that the polynomial 0 is divisible by every monic polynomial and that the monic polynomial $D(X) = 1$ divides every polynomial. If $N(X) \neq 0$, then $d = \deg[N(X)]$ is the maximum degree of the monic polynomials that divide $N(X)$. (Of course, if $N(X) = 0$, there is no maximum such degree.)

If $N_1(X)$ and $N_2(X)$ are polynomials in $F[X]$ and not both 0, then their *greatest common divisor*, which is denoted by $\gcd[N_1(X), N_2(X)]$, is the monic polynomial of greatest degree that divides both $N_1(X)$ and $N_2(X)$. We will tacitly assume that $N_1(X)$ and $N_2(X)$ are not both 0 whenever we write $\gcd[N_1(X), N_2(X)]$.

Fundamental Property of Greatest Common Divisors in F[X]: For any polynomial $I(X)$,

$$\gcd[N_1(X) + I(X)N_2(X), N_2(X)] = \gcd[N_1(X), N_2(X)], \tag{7.65}$$

i.e., adding a multiple of one polynomial to another does not change their greatest common divisor.

It is then a simple step to a "Euclidian" recursion.

Recursion of the Greatest Common Divisor in F[X]: If $N_2(X) \neq 0$, then

$$\gcd[N_1(X), N_2(X)] = \gcd[N_2(X), R_{N_2(X)}[N_1(X)]]. \tag{7.66}$$

From this recursion, one can easily design the "Euclidian" algorithm for computing $\gcd[N_1(X), N_2(X)]$ and one can then "extend" this algorithm to an algorithm that also finds the polynomials $A(X)$ and $B(X)$ in the following theorem.

Greatest Common Divisor Theorem for F[X]: For any polynomials $N_1(X)$ and $N_2(X)$, not both 0, there exist polynomials $A(X)$ and $B(X)$ such that

$$\gcd[N_1(X), N_2(X)] = A(X)N_1(X) + B(X)N_2(X). \tag{7.67}$$

Remark: The polynomials $A(X)$ and $B(X)$ specified in this theorem are *not* unique.

The reader might now suspect that one can make a ring of polynomials modulo a given polynomial and that, under certain conditions on the modulus, the ring will be a field. This is indeed the case and will be carried out in the exercises.

Chapter 8

RUDIMENTS OF ALGEBRAIC CODING THEORY

8.1 Introduction

Shannon's statement and proof of the Noisy Coding Theorem in 1948 immediately touched off a search for specific "good codes" that continues unabated today. Before long, a theory of codes began to develop that focused attention more and more on the algebraic structure of codes and less and less on the role of codes in the Noisy Coding Theorem. Within two decades after Shannon's work appeared, "algebraic coding theory" had already established itself as a scientific field in its own right. The results and techniques of this field soon began to find applications to many problems beyond channel coding; indeed, the relevance of many algebraic coding techniques to actual channel coding problems is at best tenuous.

Our goal in this chapter is to introduce the main ideas of algebraic coding theory in such a way that the interested reader will be well positioned to explore these ideas further in the rather vast literature of this field.

8.2 Block Codes, Dimensionless Rate and Encoders

We will confine our attention in this chapter to codes in which the code digits take values in a finite field $GF(q)$, which is the most important case both theoretically and practically, although some of the results that we derive are valid when the code digits take values in an arbitrary finite alphabet.

By a *q-ary block code of length N*, we will mean any non-empty subset of the vector space of N-tuples ("row vectors") $GF(q)^N$. It is important to note that a block code, as defined in algebraic coding theory, is a *set* (and *not a list* as was the case when we considered the Noisy Coding Theorem), i.e., the codewords must all be different. Thus, the number M of codewords in a q-ary block code of length N must satisfy

$$1 \leq M \leq q^N. \tag{8.1}$$

By the *dimensionless rate* of a q-ary block code of length N, we will mean the number

$$R = \frac{\log_q M}{N}, \tag{8.2}$$

where M is the number of codewords [and *not* $(\log_2 M)/N$, which is the "dimensioned rate" in bits/use that we considered for the Noisy Coding Theorem]. It follows from (8.1) and (8.2) that the dimensionless rate R (which we will hereafter often refer to simply as the *rate*) of a q-ary block of length N satisfies

$$0 \leq R \leq 1. \tag{8.3}$$

In most cases (and always for the case of linear codes), RN will be a positive integer that we will denote by K and will call *the number of information digits*. [Note that the code rate is then $R = K/N$.] The reason for this terminology is that we can consider the q^K K-tuples in $GF(q)^K$ as the information sequences that are mapped by an encoder onto the $M = q^{NR} = q^K$ codewords. Thus, when RN is a positive integer K, we will consider an *encoder* for a q-ary block code of length N and rate R to be an invertible (or bijective) mapping from $GF(q)^K$ to the q^K codewords in $GF(q)^N$. This is illustrated in Fig. 8.1, where $\mathbf{a} = [a_1, a_2, \ldots, a_K]$ is the information sequence in $GF(q)^K$ and $\mathbf{b} = [b_1, b_2, \ldots, b_N]$ is the codeword in $GF(q)^N$. It is important to note that the *same code has many encoders*. In fact, if we order the q^K codewords in some way, there are q^K choices for the information sequence assigned to the first codeword, $q^K - 1$ choices then for the information sequence assigned to the second codeword, etc. Thus, there are exactly $(q^K)!$ different encoders for a given q-ary block code of length N and rate R when $K = RN$ is a positive integer.

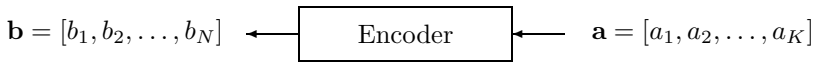

Figure 8.1: An encoder for a q-ary block code of length N with dimensionless rate K/N.

8.3 Hamming Weight and Distance

The *Hamming distance*, $d(\mathbf{x}, \mathbf{y})$, between N-tuples $\mathbf{x}$ and $\mathbf{y}$ with components in an arbitrary non-empty set is defined as the number of coordinates in which $\mathbf{x}$ and $\mathbf{y}$ differ. It is easy to see that Hamming distance is a metric for the set of N-tuples over any non-empty set, i.e., that it satisfies the following three axioms for any three N-tuples $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ with components in this set:

(M1) $d(\mathbf{x}, \mathbf{y}) \geq 0$ with equality if and only if $\mathbf{x} = \mathbf{y}$ (positive definiteness);

(M2) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ (symmetry); and

(M3) $d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}) \geq d(\mathbf{x}, \mathbf{y})$ (triangle inequality).

That (M3) holds for Hamming distance follows from the fact that in any coordinate where $\mathbf{x}$ and $\mathbf{y}$ differ it must be true that either $\mathbf{x}$ and $\mathbf{z}$ differ or that $\mathbf{z}$ and $\mathbf{y}$ differ (or both).

The *Hamming weight*, $w(\mathbf{x})$, of a vector $\mathbf{x}$ in F^N is defined to be the number of coordinates in which $\mathbf{x}$ is non-zero. It follows that

$$d(\mathbf{x}, \mathbf{y}) = w(\mathbf{y} - \mathbf{x}), \tag{8.4}$$

since $\mathbf{x}$ and $\mathbf{y}$ will differ in some coordinate if and only if $\mathbf{y}-\mathbf{x}$ is non-zero in that coordinate. The following three properties of Hamming weight are the direct analogs of the axioms (M1), (M2) and (M3) of a metric. For any $\mathbf{x}$ and $\mathbf{y}$ in F^N,

(W1) $w(\mathbf{x}) \geq 0$ with equality if and only if $\mathbf{x} = \mathbf{0}$;

(W2) $w(\mathbf{x}) = w(-\mathbf{x})$; and

(W3) $w(\mathbf{x}) + w(\mathbf{y}) \geq w(\mathbf{x}+\mathbf{y})$.

The *Hamming sphere of radius* r [where r is an integer satisfying $0 \leq r \leq N$] centered at some $\mathbf{x}$ in $GF(q)^N$ is the set $S_r(\mathbf{x})$ of all $\mathbf{y}$ in $GF(q)^N$ such that $d(\mathbf{x}, \mathbf{y}) \leq r$. The *volume* of the sphere $S_r(\mathbf{x})$ is the number of N-tuples in $S_r(\mathbf{x})$, $\#S_r(\mathbf{x})$. From the fact that there are $\binom{N}{i}$ ways to choose the i coordinates in which $\mathbf{y}$ differs from $\mathbf{x}$ and $q-1$ ways to choose a component for $\mathbf{y}$ different from the component of $\mathbf{x}$ in each of these i coordinates, it follows that the volume V_r of the Hamming sphere $S_r(\mathbf{x})$ does not depend on $\mathbf{x}$ and is given by

$$V_r = \sum_{i=0}^{r} \binom{N}{i} (q-1)^i. \tag{8.5}$$

8.4 Error Correction and Detection

Suppose that a codeword $\mathbf{b}$ in a q-ary block code B of length N and rate R is transmitted via some "noisy channel" and that $\mathbf{r}$ is the received word. The usual assumption of algebraic coding theory, and the one that we shall make here, is that $\mathbf{r}$ is also an N-tuple in $GF(q)^N$. Then the *error pattern*, $\mathbf{e}$, that occurred in this transmission is defined as

$$\mathbf{e} = \mathbf{r} - \mathbf{b}. \tag{8.6}$$

Each component of $\mathbf{r}$ that is different from the corresponding component of $\mathbf{b}$ represents an "error" in transmission. Thus, the *number of errors* that occur is defined as $d(\mathbf{b}, \mathbf{r})$ or, equivalently, as $w(\mathbf{e})$.

By a *decoder* for a q-ary block code B of length N and rate R, we will mean a mapping $f(\cdot)$ from $GF(q)^N$ to B. The interpretation is that $f(\mathbf{r})$ is the decoder's decision for the transmitted codeword when $\mathbf{r}$ is the received word.

We will say that the code B *can correct all the error patterns in a set* E (which is a non-empty subset of $GF(q)^N$) if there is a decoding function $f(\cdot)$ for B such that

$$f(\mathbf{b}+\mathbf{e}) = \mathbf{b} \qquad \text{all } \mathbf{b} \in B, \text{ all } \mathbf{e} \in E. \tag{8.7}$$

The interpretation of (8.7) is that the decoding decision made by the decoder $f(\cdot)$ will always be corrected when a codeword of B is transmitted and an error pattern in E occurs.

The *minimum distance* of a q-ary block code of length N and rate R, denoted $d_{\min}$, is defined when $R > 0$ as the smallest Hamming distance between two distinct codewords; when $R = 0$ so that there is only $M = q^{NR} = 1$ codeword, one says by way of convention that $d_{\min} = \infty$.

Error-Correcting Capability of a Code: A q-ary block code of length N and minimum distance $d_{\min}$ can correct all patterns of t or fewer errors (where t is an integer $0 \leq t \leq N$) if and only if

$$d_{\min} > 2t.$$

Proof: Suppose the code cannot correct all patterns of t or fewer errors. Then, because (8.7) cannot be satisfied for all codewords, there must be distinct codewords $\mathbf{b}$ and $\mathbf{b}'$ together with error patterns $\mathbf{e}$ and $\mathbf{e}'$, both of Hamming weight t or less, such that $\mathbf{b}+\mathbf{e} = \mathbf{b}'+\mathbf{e}'$. Thus $\mathbf{b}-\mathbf{b}' = \mathbf{e}'-\mathbf{e}$ and hence $d(\mathbf{b}', \mathbf{b}) = w(\mathbf{b}-\mathbf{b}') = w(\mathbf{e}'-\mathbf{e}) \leq w(\mathbf{e}') + w(-\mathbf{e}) = w(\mathbf{e}') + w(\mathbf{e}) \leq 2t$, where we have made use of (W3) and (W2). It follows that $d_{\min} \leq 2t$.

Conversely, suppose $d_{\min} \leq 2t$. Then there exist distinct codewords $\mathbf{b}$ and $\mathbf{b}'$ with $d(\mathbf{b}, \mathbf{b}') = d_{\min} = w(\mathbf{b}-\mathbf{b}') \leq 2t$. But any vector $\mathbf{b}-\mathbf{b}'$ of Hamming weight at most $2t$ can be written as the difference $\mathbf{e}'-\mathbf{e}$ of two vectors each of Hamming weight at most t [just assign the first non-zero component of $\mathbf{b}-\mathbf{b}'$ to $\mathbf{e}'$, the second to $-\mathbf{e}$, the third to $\mathbf{e}'$, etc.. Hence, $\mathbf{b}-\mathbf{b}' = \mathbf{e}'-\mathbf{e}$ or $\mathbf{b}+\mathbf{e} = \mathbf{b}'+\mathbf{e}'$. It now follows that (8.7) cannot be satisfied for both $\mathbf{b}$ and $\mathbf{b}'$ when E is the set of all error patterns of weight t or less. Thus, not all patterns of weight t or less can be corrected. □

The previous result, although specifying the error-correcting capability of a code, does not quite tell us how to build the corresponding decoder.

Correcting t or Fewer Errors: A decoder $f(\cdot)$ for a code with $d_{\min} > 2t$ will correct all patterns of t or fewer errors if and only if $f(\mathbf{r})$ is the codeword nearest to $\mathbf{r}$ in Hamming distance for all $\mathbf{r}$ such that the nearest codeword $\mathbf{b}$ satisfies $d(\mathbf{b}, \mathbf{r}) \leq t$.

Proof: If $\mathbf{b}$ is the transmitted codeword and $\mathbf{r}$ is the received word, then t or fewer errors have occurred if and only if $d(\mathbf{b}, \mathbf{r}) \leq t$, i.e., if and only if $\mathbf{r}$ lies in the Hamming sphere $S_t(\mathbf{b})$ of radius t centered at $\mathbf{b}$. Thus, a decoder that corrects all patterns of t or fewer errors must decode every $\mathbf{r}$ in $S_t(\mathbf{b})$ to $\mathbf{b}$. But if $\mathbf{r}$ is in $S_t(\mathbf{b})$ and $\mathbf{b}'$ is a codeword distinct from $\mathbf{b}$, then the triangle inequality (M3) implies $d(\mathbf{b}', \mathbf{r}) \geq d(\mathbf{b}, \mathbf{b}') - d(\mathbf{r}, \mathbf{b}) \geq d_{\min} - d(\mathbf{r}, \mathbf{b}) \geq d_{\min} - t > 2t - t = t$, where we have used the fact that, because the code corrects all patterns of t or fewer errors, $d_{\min} > 2t$. Thus, every $\mathbf{r}$ in $S_t(\mathbf{b})$ is strictly closer to $\mathbf{b}$ in Hamming distance than to any other codeword. □

Sometimes one is content to "detect" certain types of errors without trying to correct these errors. This is often the case in various types of Automatic Repeat reQuest (ARQ) systems where the receiver can request a retransmission of a codeword if it is unable to make a reliable decision. To describe this situation, we define an *incomplete decoder* for a q-ary block code B of length N to be a mapping $f(\cdot)$ from $GF(q)^N$ to $B \cup \{\mathbf{?}\}$. The interpretation of $f(\mathbf{r}) = \mathbf{?}$ is that the incomplete decoder has detected an error but is unwilling to try to correct this error. We will say that the code B can correct all the error patterns in a *non-empty* set E_1 and detect all the error patterns in a set E_2, where $E_1 \cap E_2 = \varnothing$, if there is an incomplete decoding function f for B such that

$$f(\mathbf{b}+\mathbf{e}) = \begin{cases} \mathbf{b} & \text{all } \mathbf{b} \in B \text{ and all } \mathbf{e} \in E_1 \\ \mathbf{?} & \text{all } \mathbf{b} \in B \text{ and all } \mathbf{e} \in E_2. \end{cases} \tag{8.8}$$

If $E_1 = \{\mathbf{0}\}$, then one says that the code is being used *purely for error detection.*

Error-Detecting and Correcting Capability of a Code: A q-ary block code of length N and minimum distance $d_{\min}$ can correct all patterns of t or fewer errors and detect all patterns of $t+1, t+2, \ldots, t+s$ errors, where $0 \leq t < t+s \leq N$, if and only if

$$d_{\min} > 2t + s.$$

Proof: Suppose that the code cannot correct and detect as specified. Then, because (8.8) cannot be satisfied for all codewords, there must be distinct codewords $\mathbf{b}$ and $\mathbf{b}'$, together with error patterns $\mathbf{e}$

and $\mathbf{e}'$ satisfying $w(\mathbf{e}) \leq t$ and $w(\mathbf{e}') \leq t+s$, such that $\mathbf{b}+\mathbf{e} = \mathbf{b}'+\mathbf{e}'$. Thus, $d(\mathbf{b}', \mathbf{b}) = w(\mathbf{b}-\mathbf{b}') = w(\mathbf{e}'-\mathbf{e}) \leq w(\mathbf{e}') + w(-\mathbf{e}) = w(\mathbf{e}') + w(\mathbf{e}) \leq t+s+t = 2t+s$. It follows that $d_{\min} \leq 2t+s$.

Conversely, suppose that $d_{\min} \leq 2t+s$. Then there exist distinct codewords $\mathbf{b}$ and $\mathbf{b}'$ with $d(\mathbf{b}', \mathbf{b}) = w(\mathbf{b}-\mathbf{b}') = d_{\min} \leq 2t+s$. But any vector $\mathbf{b}-\mathbf{b}'$ of Hamming weight at most $2t+s$ can be written as the difference $\mathbf{e}'-\mathbf{e}$ of error patterns $\mathbf{e}$ and $\mathbf{e}'$ with $w(\mathbf{e}) \leq t$ and $w(\mathbf{e}') \leq t+s$ [just assign the first t non-zero components of $\mathbf{b}'-\mathbf{b}$ to $-\mathbf{e}$ (or all components if $w(\mathbf{b}-\mathbf{b}') < t$) and the remaining non-zero components of $\mathbf{b}'-\mathbf{b}$ to $\mathbf{e}'$]. Then

$$\mathbf{b}+\mathbf{e} = \mathbf{b}'+\mathbf{e}'$$

where $\mathbf{b}$ and $\mathbf{b}'$ are distinct codewords, $\mathbf{e}$ is a pattern of t or fewer errors, and $\mathbf{e}'$ is a pattern of $t+s$ or fewer errors. Thus (8.8) cannot be satisfied for both $\mathbf{b}$ and $\mathbf{b}'$ since this would require $f(\mathbf{b}+\mathbf{e})$ to equal $\mathbf{b}$ and at the same time $f(\mathbf{b}'+\mathbf{e}') = f(\mathbf{b}+\mathbf{e})$ to equal either $\mathbf{b}'$ or **?**. It follows that not all patterns of t or fewer errors can be corrected and all patterns of $t+1, t+2, \ldots, t+s$ errors detected. □

The above results about error correction and error correction-and-detection should make it clear why the minimum distance $d_{\min}$ of a code is of central importance in algebraic coding theory.

8.5 Linear Codes

We now define the class of codes that are of primary interest in algebraic coding theory. An (N, K) *q-ary linear code* (or simply an (N, K) *q-ary code*) is a K-dimensional subspace of the vector space $GF(q)^N$ of N tuples over $GF(q)$.

If $K > 0$ and if $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_K$ are a basis for the (N, K) q-ary linear code $\mathbf{V}$, then every codeword $\mathbf{b}$ can be written uniquely as a linear combination

$$\mathbf{b} = a_1\mathbf{g}_1 + a_2\mathbf{g}_2 + \cdots + a_k\mathbf{g}_K \tag{8.9}$$

where $a_1, a_2, \ldots, a_k$ are elements of the scalar field $GF(q)$; conversely, every such linear combination is a codeword. It follows that there are exactly q^K codewords and hence that the code rate is

$$R = \frac{K}{N}. \tag{8.10}$$

Note that (8.9) describes a *linear encoding rule* for encoding the information vector $\mathbf{a} = [a_1, a_2, \ldots, a_K]$ in $GF(q)^K$ to the codeword $\mathbf{b}$ in $GF(q)^N$. We can write (8.9) more compactly as

$$\mathbf{b} = \mathbf{a}\,G \tag{8.11}$$

where G is the matrix whose rows are $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_K$. We shall call any matrix G whose rows are a basis for the linear code $\mathbf{V}$ an *encoding matrix for* $\mathbf{V}$ [but we warn the reader that the terminology *generator matrix for* $\mathbf{V}$ is more usual in the literature]. If $\mathbf{V}$ is an (N, K) q-ary linear code with $K > 0$, then an encoding matrix for $\mathbf{V}$ is a $K \times N$ matrix whose $K{\cdot}N$ entries are in $GF(q)$ and whose rows are K linearly independent codewords.

An (N, K) q-ary linear code with $K > 0$ has many encoding matrices. Any of the $q^K - 1$ non-zero codewords can be chosen as the first row $\mathbf{g}_1$ of G. Any of the $q^K - q$ codewords not in $S(\mathbf{g}_1)$ can be chosen as the second row $\mathbf{g}_2$ of G. Any of the $q^K - q^2$ codewords not in $S(\mathbf{g}_1, \mathbf{g}_2)$ can be chosen as the

third row $\mathbf{g}_3$ of G. Etc. It follows that an (N, K) q-ary linear code has exactly

$$\prod_{i=0}^{K-1} (q^K - q^i)$$

distinct encoding matrices, i.e., distinct linear encoders. Although usually a vast number, the number of linear encoders for $\mathbf{V}$ is nonetheless dwarfed by the number $(q^K)!$ of general encoders for $\mathbf{V}$ that we determined in Section 8.2. The advantage of the linear encoders is that they are very easy to implement.

Example 8.5.1 The binary vectors $[1, 0, 1], [0, 1, 1]$ are linearly independent and hence form a basis, $\mathbf{g}_1, \mathbf{g}_2$, for a $(3, 2)$ binary code. The corresponding encoding matrix is

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

The linear encoding equation (8.11) gives the codeword $\mathbf{b} = [b_1, b_2, b_3]$ corresponding to the information vector $\mathbf{a} = [a_1, a_2]$ as

$$\begin{aligned} b_1 &= a_1 \\ b_2 &= a_2 \\ b_3 &= a_1 + a_2. \end{aligned}$$

Note that this encoder can be implemented with a single $GF(2)$ adder, i.e., with a single exclusive-OR (XOR) gate. [The reader should reflect why $(K-1)(N)$ such XOR gates suffice to implement any linear encoder for any (N, K) binary code with $K > 0$]. This $(3, 2)$ binary code is the vector space $\mathbf{V} = \{[0, 0, 0], [1, 0, 1], [0, 1, 1], [1, 1, 0]\}$. One sees that $d_{\min} = 2$. Thus this code could be used for "single error detection".

8.6 Weight and Distance Equivalence in Linear Codes

One of the principal advantages of linear codes is that for most purposes it suffices to consider the Hamming weights of codewords where for general block codes one would have to consider Hamming distance. In other words, in linear codes it suffices to look at the codewords one at a time rather than in pairs.

The *weight enumerator* of an (N, K) q-ary linear code is the list $(A_0, A_1, \ldots, A_N)$ where A_i is the number of codewords of Hamming weight i. Note that the A_i must be nonnegative integers and that $A_0 = 1$ and $A_0 + A_1 + \cdots + A_N = q^K$. The *minimum weight* of an (N, K) q-ary linear code, denoted $w_{\min}$, is the smallest Hamming weight of the non-zero codewords (and by way of convention is ∞ in case $K = 0$ so that $\mathbf{0}$ is the only codeword.) Equivalently, $w_{\min}$ is the smallest positive integer i such that $A_i \neq 0$.

Weight and Distance Equivalence in Linear Codes: For every integer $i, 0 \leq i \leq N$, and every choice of a codeword $\mathbf{b}$ in an (N, K) q-ary linear code $\mathbf{V}$, the number of codewords at Hamming distance i from $\mathbf{b}$ is equal to the number A_i of codewords having Hamming weight i.

Proof: For every choice of a codeword $\mathbf{b}$ in $\mathbf{V}$, we have

$$\begin{aligned}\#\{\mathbf{b}' : \mathbf{b}' \in \mathbf{V} \text{ and } d(\mathbf{b}, \mathbf{b}') = i\} &= \#\{\mathbf{b}' : \mathbf{b}' \in \mathbf{V} \text{ and } w(\mathbf{b}' - \mathbf{b}) = i\} \\ &= \#\{\mathbf{b}' - \mathbf{b} : \mathbf{b}' \in \mathbf{V} \text{ and } w(\mathbf{b}' - \mathbf{b}) = i\} \\ &= \#\{\mathbf{b}'' : \mathbf{b}'' \in \mathbf{V} - \mathbf{b} \text{ and } w(\mathbf{b}'') = i\} \\ &= \#\{\mathbf{b}' : \mathbf{b}' \in \mathbf{V} \text{ and } w(\mathbf{b}') = i\} \\ &= A_i\end{aligned}$$

where we have used the fact that, because $\mathbf{b}$ is a codeword, i.e., a member of the additive group $\langle \mathbf{V}, + \rangle$, as is also $-\mathbf{b}$, the coset $\mathbf{V} - \mathbf{b}$ coincides with $\mathbf{V}$. □

As an immediate corollary of this equivalence, we have the following very important property of linear codes.

Equality of $\mathbf{d_{min}}$ and $\mathbf{w_{min}}$: In an (N, K) q-ary linear code, $d_{\min} = w_{\min}$.

8.7 Orthogonality in F^N

In this section, we consider certain general properties of F^N for an arbitrary field F that we will apply in the next section to linear codes.

Two vectors $\mathbf{u} = [u_1, u_2, \ldots, u_N]$ and $\mathbf{v} = [v_1, v_2, \ldots, v_N]$ in F^N will be called *orthogonal* if their "dot product" vanishes, i.e., if $\mathbf{u} \bullet \mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_N v_N = 0$. Note that because we always consider vectors in F^N to be "row vectors" (i.e., $1 \times N$ matrices), we can write the dot product conveniently as $\mathbf{u} \bullet \mathbf{v} = \mathbf{u}\mathbf{v}^T$, where the superscript T denotes transpose. Thus, $\mathbf{u}$ and $\mathbf{v}$ are orthogonal just when $\mathbf{u}\mathbf{v}^T = 0$.

Example 8.7.1 Taking $F = GF(2)$ and $\mathbf{v} = [1, 1]$, we find that $\mathbf{v}\mathbf{v}^T = 1 + 1 = 0$, i.e., the non-zero vector $\mathbf{v}$ is orthogonal to itself. In fact, if $F = GF(2)$, any vector $\mathbf{v}$ of even Hamming weight is orthogonal to itself. [This orthogonality of a non-zero vector to itself can never happen when F is the real field $\mathbb{R}$. Why not?]

If $\mathbf{V}$ is a subspace of F^N, then one writes $\mathbf{V}^\perp$ to denote the set of vectors that are orthogonal to every vector in $\mathbf{V}$, i.e.,

$$\mathbf{V}^\perp = \{\mathbf{v}' : \mathbf{v}' \in F^N \text{ and } \mathbf{v}'\mathbf{v}^T = 0 \text{ for all } \mathbf{v} \in \mathbf{V}\}.$$

If $\mathbf{v}'\mathbf{v}^T = 0$ then $(c\mathbf{v}')\mathbf{v}^T = c(\mathbf{v}'\mathbf{v}^T) = 0$ holds for all $c \in F$. Similarly, if $\mathbf{v}_1\mathbf{v}^T = \mathbf{v}_2\mathbf{v}^T = 0$, then $(\mathbf{v}_1 + \mathbf{v}_2)\mathbf{v}^T = 0$. It follows from the test for a subspace that $\mathbf{V}^\perp$ is also a subspace of F^N. We will call $\mathbf{V}^\perp$ the *orthogonal subspace* of $\mathbf{V}$ in F^N.

If A is an $r \times N$ matrix with entries in F, then by the *row space of* A, denoted $R(A)$, one means the subspace of F^N spanned by the rows of A. The dimension of the subspace $R(A)$ is of course at most r, the number of rows of A, with equality if and only if the rows of A are linearly independent vectors in F^N.

Matrix Characterization of Orthogonal Subspaces: If $\mathbf{V}$ is a subspace of F^N and if A is a matrix with entries in F such that $\mathbf{V} = R(A)$, then a vector $\mathbf{v}'$ in F^N lies in $\mathbf{V}^\perp$ if and only if $\mathbf{v}'A^T = \mathbf{0}$.

Proof: Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ be the rows of A and note that $\mathbf{v}'A^T = \mathbf{v}'[\mathbf{v}_1^T \cdots \mathbf{v}_r^T] = [\mathbf{v}'\mathbf{v}_1^T, \ldots, \mathbf{v}'\mathbf{v}_r^T]$. Suppose $\mathbf{v}'$ lies in $\mathbf{V}^\perp$, then $\mathbf{v}'$ is orthogonal to each row of A so that certainly $\mathbf{v}'A^T = \mathbf{0}$.

Conversely, suppose $\mathbf{v}'A^T = \mathbf{0}$. Then $\mathbf{v}'\mathbf{v}_i^T = 0$ for $i = 1, 2, \ldots, r$. But any vector $\mathbf{v}$ in $\mathbf{V} = R(A)$ can be written as $\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_r\mathbf{v}_r$. Hence $\mathbf{v}'\mathbf{v}^T = \mathbf{v}'(c_1\mathbf{v}_1^T + c_2\mathbf{v}_2^T + \cdots + c_r\mathbf{v}_r^T) = c_1\mathbf{v}'\mathbf{v}_1^T + c_2\mathbf{v}'\mathbf{v}_2^T + \cdots + c_r\mathbf{v}'\mathbf{v}_r^T = 0$, so that $\mathbf{v}'$ is indeed orthogonal to every vector in $\mathbf{V}$. □

Fundamental Properties of Orthogonal Subspaces: If $\mathbf{V}$ is a K-dimensional subspace of F^N and $\mathbf{V}^\perp$ is its orthogonal subspace, then

$$\dim(\mathbf{V}^\perp) = N - K$$

and

$$(\mathbf{V}^\perp)^\perp = \mathbf{V}.$$

Proof: Because $F^N = \{\mathbf{0}\}^\perp$ and $\{\mathbf{0}\} = (F^N)^\perp$, the claim is trivially true if $K = 0$ or $K = N$. Suppose then that $1 \le K < N$. Let G be a $K \times N$ matrix whose rows $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K$ are a basis for $\mathbf{V}$. Let $\mathbf{v}_{K+1}, \mathbf{v}_{K+2}, \ldots, \mathbf{v}_N$ be any vectors in F^N such that $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ are a basis for F^N. [It suffices to pick $\mathbf{v}_i$ to be any vector in $F^N \setminus S(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{i-1})$ for $i = K+1, K+2, \ldots, N$.] Let $\tilde{G}$ be the $(N-K) \times N$ matrix whose rows are $\mathbf{v}_{K+1}, \mathbf{v}_{K+2}, \ldots, \mathbf{v}_N$. Then

$$M = \begin{bmatrix} G \\ \tilde{G} \end{bmatrix} \tag{8.12}$$

is an $N \times N$ matrix with linearly independent rows, i.e., an invertible matrix. Thus, there is a unique inverse matrix

$$M^{-1} = \begin{bmatrix} \tilde{H}^T & H^T \end{bmatrix} \tag{8.13}$$

where $\tilde{H}^T$ denotes the first K columns of M^{-1}. Because

$$M\ M^{-1} = \begin{bmatrix} G\tilde{H}^T & GH^T \\ \tilde{G}\tilde{H}^T & \tilde{G}H^T \end{bmatrix} = I_N = \begin{bmatrix} I_K & \mathbf{0} \\ \mathbf{0} & I_{N-K} \end{bmatrix}$$

where I_n denotes the $n \times n$ identity matrix, it follows that

$$G\tilde{H}^T = I_K \tag{8.14}$$

$$GH^T = \mathbf{0} \tag{8.15}$$

$$\tilde{G}\tilde{H}^T = \mathbf{0} \tag{8.16}$$

$$\tilde{G}H^T = I_{N-K}. \tag{8.17}$$

Because the rows of M are a basis for F^N, every $\mathbf{v} \in F^N$ can be written uniquely as $\mathbf{v} = [\mathbf{a}, \tilde{\mathbf{a}}]\, M$, i.e., as

$$\mathbf{v} = \mathbf{a}\, G + \tilde{\mathbf{a}}\, \tilde{G}.$$

But $\mathbf{v}$ is in $R(G)$ precisely when $\mathbf{v}$ can be written as $\mathbf{a}\, G$ for some $\mathbf{a}$, i.e., as $\mathbf{a}\, G + \mathbf{0}\, \tilde{G}$. Thus, it follows from the uniqueness of $\mathbf{a}$ and $\tilde{\mathbf{a}}$ that $\mathbf{v} \in \mathbf{V} = R(G)$ if and only if $\tilde{\mathbf{a}} = \mathbf{0}$. But

$$\mathbf{v}H^T = \mathbf{a}\, GH^T + \tilde{\mathbf{a}}\, \tilde{G}H^T = \tilde{\mathbf{a}}$$

where we have made use of (8.15) and (8.17). Thus, by the matrix characterization of orthogonal subspaces, $\mathbf{v} \in R(H)^\perp$ if and only if $\tilde{\mathbf{a}} = \mathbf{0}$. Thus, we have proved that

$$R(G) = R(H)^\perp. \tag{8.18}$$

Because M^{-1} is invertible, its columns are linearly independent; equivalently, the rows of the matrix

$$(M^{-1})^T = \begin{bmatrix} \tilde{H} \\ H \end{bmatrix}$$

are linearly independent. Thus, every $\mathbf{v} \in F^N$ can be written uniquely as

$$\mathbf{v} = \tilde{\mathbf{c}}\,\tilde{H} + \mathbf{c}\,H.$$

It follows from the uniqueness of $\mathbf{c}$ and $\tilde{\mathbf{c}}$ that $\mathbf{v} \in R(H)$ if and only if $\tilde{\mathbf{c}} = \mathbf{0}$. But for all $\mathbf{v} \in F^N$,

$$\begin{aligned} \mathbf{v}\,G^T &= \tilde{\mathbf{c}}\,\tilde{H}G^T + \mathbf{c}\,HG^T \\ &= \tilde{\mathbf{c}}(G\tilde{H}^T)^T + \mathbf{c}(GH^T)^T \\ &= \tilde{\mathbf{c}} \end{aligned}$$

where we have made use of (8.14) and (8.16). Thus, by the matrix characterization of orthogonal subspaces, $\mathbf{v} \in R(G)^\perp$ if and only if $\tilde{\mathbf{c}} = \mathbf{0}$. Hence, we have proved that

$$R(G)^\perp = R(H). \tag{8.19}$$

Recalling that $\mathbf{V} = R(G)$ and that the $N - K$ rows of H are linearly independent, it follows from (8.19) that $\dim(\mathbf{V}^\perp) = \dim(R(H)) = N - K$. From (8.18), it follows further that $\mathbf{V} = (\mathbf{V}^\perp)^\perp$. □

8.8 Parity-Check Matrices and Dual Codes

A *parity-check matrix* for an (N, K) q-ary linear code $\mathbf{V}$ is any matrix H with elements in $GF(q)$ such that a vector $\mathbf{v}$ in $GF(q)^N$ is a codeword (i.e., $\mathbf{v} \in \mathbf{V}$) just when $\mathbf{v}\,H^T = \mathbf{0}$. By the matrix characterization of orthogonal subspaces, it follows that *H is a parity-check matrix for $\mathbf{V}$ if and only if $\mathbf{V} = R(H)^\perp$*, i.e., if and only if the code $\mathbf{V}$ is the orthogonal subspace of the row space of H. By the Fundamental Properties of Orthogonal Subspaces, it follows further that $R(H) = \mathbf{V}^\perp$ and that $\dim(\mathbf{V}^\perp) = N - K$. The codes $\mathbf{V}$ and $\mathbf{V}^\perp$ are called *dual codes*.

A parity-check matrix H for the code $\mathbf{V}$ will be called *reduced* if its rows form a basis for $\mathbf{V}^\perp$, i.e., if this parity-check matrix is an $(N - K) \times N$ matrix. A reduced parity-check matrix for the code $\mathbf{V}$ is an encoding matrix for the dual code $\mathbf{V}^\perp$, and conversely. Our proof of the Fundamental Properties of Orthogonal Subspaces actually shows how to construct a reduced parity-check matrix.

Construction of a Reduced Parity-Check Matrix: A reduced parity-check matrix H for an (N, K) q-ary linear code with $1 \le K < N$ having an encoding matrix G (whose rows we denote here by $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K$) can be constructed as follows:

(1) Choose $\mathbf{v}_i$ as any vector in $GF(q)^N \setminus S(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{i-1})$ for $i = K+1, K+2, \ldots, N$.

(2) Form the $N \times N$ matrix M whose rows are $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$. Then compute the inverse matrix M^{-1}.

(3) Take H^T as the *last* $N-K$ columns of the matrix M^{-1}.

Example 8.8.1 Consider the $(3,2)$ binary linear code of Example 8.5.1. The encoding matrix G given there has rows $\mathbf{v}_1 = [1,0,1]$ and $\mathbf{v}_2 = [0,1,1]$. Choosing $\mathbf{v}_3 = [0,0,1]$, we obtain the upper triangular matrix

$$M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

whose inverse is itself, i.e.,

$$M^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Taking H^T as the last column of M^{-1} gives

$$H = [1,1,1].$$

Writing $\mathbf{b} = [b_1, b_2, b_3]$ we see that $\mathbf{b}$ is a codeword, i.e., $\mathbf{b}\,H^T = \mathbf{0}$, if and only if

$$b_1 + b_2 + b_3 = 0.$$

In general, each row of a parity-check matrix H gives the coefficients in a linear combination of digits that must vanish for codewords, i.e., a "*parity check*" that must be satisfied. These parity checks are all linearly independent if and only if H is a reduced parity-check matrix.

Let $\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_N^T$ be the columns of H, i.e., $H = [\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_N^T]$. Then

$$\mathbf{b}\,H^T = \mathbf{b} \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_N \end{bmatrix} = b_1\mathbf{c}_1 + b_2\mathbf{c}_2 + \cdots + b_N\mathbf{c}_N.$$

Because $\mathbf{b}\,H^T = \mathbf{0}$ if $\mathbf{b}$ is a non-zero codeword, it follows that a codeword $\mathbf{b}$ of weight w determines a vanishing linear combination with all coefficients non-zero of w *columns* of H (rows of H^T), and conversely. It follows that the minimum weight $w_{\min}$ of the code equals the smallest such n, and hence so also does $d_{\min}$. Thus, we have proved the following useful result.

Determination of Minimum Distance from a Parity-Check Matrix: If H is any parity-check matrix for an (N,K) q-ary linear code with $1 \le K < N$, then the minimum distance $d_{\min}$ of this code equals the smallest positive integer n such that there are n columns of H that are linearly dependent.

Example 8.8.2 Consider the parity-check matrix

$$H = [1,1,1]$$

found in Example 8.8.1 for the $(3,2)$ binary linear code of Example 8.5.1. No single column of H is linearly dependent (i.e., no column of H is the all-zero column). However, there is a pair of linearly dependent columns (and, in fact, every pair of columns of H is linearly dependent for this special code). Thus, $d_{\min} = 2$.

An (N, K) q-ary linear code is called *systematic* if it has an encoding matrix of the form $G = [I_K \;\; P]$ where P is some $K \times (N-K)$ matrix; if it exists, such an encoding matrix is easily seen to be unique and is called *the systematic encoding matrix*. If the systematic encoding matrix is used for encoding, then the information vector $\mathbf{a}$ gives the codeword

$$\begin{aligned}\mathbf{b} &= \mathbf{a}[I_K \;\; P] \\ &= [\mathbf{a}\, I_K, \mathbf{a}\, P] \\ &= [\mathbf{a}, \mathbf{a}\, P]\end{aligned}$$

in which $\mathbf{a}$ appears unchanged as the first K digits. This, in fact, is why the encoding is called "systematic". Finding a parity-check matrix is especially easy when the code is systematic and one knows the systematic encoding matrix.

Parity-Check Matrices for Systematic Codes: If $\mathbf{V}$ is a systematic (N, K) q-ary linear code with $1 \leq K \leq N$ and if $G = [I_K \;\; P]$ is its systematic encoding matrix, then $H = [-P^T \;\; I_{N-K}]$ is a reduced parity-check matrix for $\mathbf{V}$.

Proof: We may choose the matrix M in our general construction of reduced parity-check matrices as the upper triangular matrix

$$M = \begin{bmatrix} I_K & P \\ 0 & I_{N-K} \end{bmatrix}.$$

The inverse matrix is the upper triangular matrix

$$M^{-1} = \begin{bmatrix} I_K & -P \\ 0 & I_{N-K} \end{bmatrix}$$

as the reader can directly verify by multiplying M by M^{-1}. The last $N-K$ columns of M^{-1} give H^T, the transpose of the desired reduced parity-check matrix. Thus, $H = [-P^T \;\; I_{N-K}]$ as claimed. □

Suppose $\mathbf{V}$ is any (N, K) q-ary linear code with $K \geq 1$. Then the subset of codewords of $\mathbf{V}$ that have 0 as their first component is easily seen (by the subspace test) to be a subspace of $\mathbf{V}$. Moreover, an N-tuple $\mathbf{b}$ is a codeword in this subspace if and only if $\mathbf{b}\, H^T = 0$ and $\mathbf{b}\, [1, 0, \ldots, 0]^T = \mathbf{0}$ where H is any parity-check matrix for $\mathbf{V}$. Thus, this subspace of codewords is itself a linear code $\mathbf{V}_1$ with parity-check matrix

$$H_1 = \begin{bmatrix} & H & \\ 1 & 0 \;\cdots\; & 0 \end{bmatrix}.$$

Continuing in this manner,we see that

$$H_n = \begin{bmatrix} H & \\ I_n & 0 \end{bmatrix}$$

is a parity-check matrix of the linear code $\mathbf{V}_n$ consisting of the codewords in $\mathbf{V}$ whose first n components are all 0's. If we take H to be a reduced parity-check matrix for $\mathbf{V}$, then H_n has exactly $N - K + n$ rows. Thus, $\dim(R(H_n)) \leq N - K + n$ or, equivalently (since H_n is a parity-check matrix for $\mathbf{V}_n$), $\dim(\mathbf{V}_n) \geq K - n$. It follows that $\mathbf{V}_n$ must contain non-zero codewords if $n = K - 1$. We have made the argument for the first n components of codewords, but it should be obvious to the reader that it holds for every choice of n components. We have proved the following interesting result.

Zero-Components of Codewords: For any (N, K) q-ary linear code with $K \geq 2$ and any choice of $K-1$ among the N components, there exist non-zero codewords that contain only zeroes in these components. In particular,

$$w_{\min} = d_{\min} \leq N - K + 1. \tag{8.20}$$

The upper bound (8.20) on minimum distance, which trivially holds also for $K = 1$, is called *Singleton's bound* after its discoverer. An (N, K) q-ary linear code that meets this bound with equality is called a *maximum-distance-separable* (MDS) code. The $(3, 2)$ binary linear code of Example 8.5.1 with $d_{\min} = 2$ is an MDS code, as indeed are all the (N, K) q-ary codes with $K = N - 1$ and $d_{\min} = 2$ constructed in Problem 8.1. For any N and any q, the $(N, 1)$ q-ary linear code with $G = [1\ 1 \cdots 1]$ has $d_{\min} = N$ and is thus also an MDS code. More interestingly, the Reed-Solomon codes that we shall soon study are MDS codes.

8.9 Cosets and Syndromes

We first give an algebraic characterization of the condition for a linear code to be able to correct all the error patterns in a set E, as we considered in Section 8.4. Because an (N, K) q-ary linear code $\mathbf{V}$ is a subgroup of the additive group of the vector space $GF(q)^N$, one can form the coset $\mathbf{e}+\mathbf{V}$ of $GF(q)^N$ relative to the subgroup $\mathbf{V}$ that contains a particular vector $\mathbf{e}$. These are always the "cosets" that we shall mean in this section.

Algebraic Condition for Error Correction: An (N, K) q-ary linear code $\mathbf{V}$ can correct all error patterns in a set E if and only if these error patterns all lie in different cosets of $GF(q)^N$ relative to $\mathbf{V}$.

Proof: Suppose that $\mathbf{e}$ and $\mathbf{e}'$ are distinct error patterns in E but that $\mathbf{e}+\mathbf{V} = \mathbf{e}'+\mathbf{V}$. Then for any codeword $\mathbf{b}$ there is a codeword $\mathbf{b}'$ different from $\mathbf{b}$ such that $\mathbf{b}+\mathbf{e} = \mathbf{b}'+\mathbf{e}'$. Thus, (8.7) cannot be satisfied, i.e., the code cannot correct all errors in E.

Conversely, suppose that all error patterns in E lie in different cosets relative to $\mathbf{V}$. If $\mathbf{e}$ is the actual error pattern and $\mathbf{b}$ the actual transmitted codeword, then the received word $\mathbf{r} = \mathbf{b}+\mathbf{e}$ lies in the coset $\mathbf{e}+\mathbf{V}$. Thus, all error patterns in E can be corrected by a decoder $f(\cdot)$ that maps $\mathbf{r}$ into the codeword $\mathbf{r}-\hat{\mathbf{e}}$ where $\hat{\mathbf{e}}$ is the error pattern (if any) in E that lies in the coset $\mathbf{r}+\mathbf{V}$. □

If $\mathbf{b}$ is the transmitted codeword, $\mathbf{e}$ the actual error pattern and $\mathbf{r} = \mathbf{b}+\mathbf{e}$ the received word in an (N, K) q-ary linear code $\mathbf{V}$ with parity-check matrix H, then

$$\mathbf{s} = \mathbf{r}\,H^T \tag{8.21}$$

is called the *syndrome of* $\mathbf{r}$ *relative to the parity-check matrix* H (or simply the *syndrome of* $\mathbf{r}$ for short). Note that

$$\begin{aligned} \mathbf{s} &= (\mathbf{b}+\mathbf{e})H^T \\ &= \mathbf{b}\,H^T+\mathbf{e}\,H^T \\ &= \mathbf{0}+\mathbf{e}\,H^T \\ &= \mathbf{e}\,H^T, \end{aligned}$$

which illustrates the important fact that *the syndrome of the received word depends only on the actual error pattern* and not at all on the transmitted codeword.

Equivalence of Syndromes and Cosets: The error patterns $\mathbf{e}$ and $\mathbf{e}'$ lie in the same coset of $GF(q)^N$ relative to $\mathbf{V}$, an (N, K) linear code with parity-check matrix H, if and only if $\mathbf{e}$ and $\mathbf{e}'$ have the same syndrome relative to H.

Proof: Suppose $\mathbf{e}$ and $\mathbf{e}'$ have the same syndrome, i.e., that $\mathbf{e}\,H^T = \mathbf{e}'H^T$. Then $(\mathbf{e}-\mathbf{e}')H^T = \mathbf{0}$ so that $\mathbf{e}-\mathbf{e}'$ is a codeword. Thus $\mathbf{e}-\mathbf{e}'+\mathbf{V} = \mathbf{V}$ or, equivalently, $\mathbf{e}+\mathbf{V} = \mathbf{e}'+\mathbf{V}$ so that $\mathbf{e}$ and $\mathbf{e}'$ lie in the same coset relative to $\mathbf{V}$.

Conversely, suppose that $\mathbf{e}$ and $\mathbf{e}'$ lie in the same coset, i.e., that $\mathbf{e}+\mathbf{V} = \mathbf{e}'+\mathbf{V}$. Then $\mathbf{e}-\mathbf{e}'+\mathbf{V} = \mathbf{V}$ so that $\mathbf{e}-\mathbf{e}'$ is a codeword. Thus, $(\mathbf{e}-\mathbf{e}')H^T = \mathbf{0}$ or, equivalently, $\mathbf{e}\,H^T = \mathbf{e}'H^T$ so that $\mathbf{e}$ and $\mathbf{e}'$ have the same syndrome relative to H. □

Corollary to the Algebraic Condition for Error Correction: An (N, K) q-ary linear code with parity-check matrix H can correct all error patterns in a set E if and only if the syndromes of these error patterns relative to H are all different.

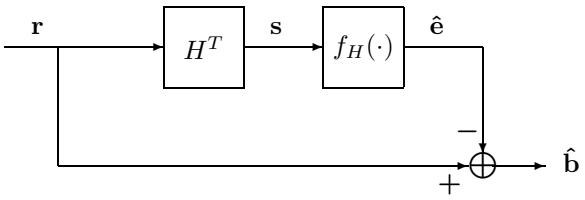

Figure 8.2: A syndrome decoder for a linear code with parity-check matrix H

This corollary suggests what is known as a *syndrome decoder* for a linear code and is shown in Fig. 8.2. The received word $\mathbf{r}$ is first mapped to the syndrome $\mathbf{s}$ of $\mathbf{r}$ relative to H^T. Then the function $f_H(\cdot)$ [the so-called *error-pattern detector*] emits that error pattern $\hat{\mathbf{e}} = f_H(\mathbf{s})$ in the coset $\mathbf{r}+\mathbf{V} = \mathbf{e}+\mathbf{V}$ that the designer has chosen to correct, i.e., to take as his decision for the actual error patten $\mathbf{e}$. The decision $\hat{\mathbf{e}}$ is then subtracted from $\mathbf{r}$ to obtain the decision $\hat{\mathbf{b}}$ for the actual transmitted word $\mathbf{b}$. It follows from the above corollary that *any set of error patterns that can be corrected by any decoder for a linear code* $\mathbf{V}$ *can also be corrected by some syndrome decoder for* $\mathbf{V}$. Almost all decoders used in practice are syndrome decoders.

8.10 Varshamov's Bound

We now consider a very interesting construction for linear codes, due to Varshamov, which is the most powerful general construction known. The idea is to build a parity-check matrix H for a q-ary linear code with minimum distance $d_{\min} \geq d$ by ensuring that no column of H is a linear combination of $d-2$ or fewer columns so that all choices of $d-1$ or fewer columns of H are linearly independent.

Suppose that $r = N - K$ is the desired redundancy of the q-ary linear code and that d is the desired (lower bound on) minimum distance, where $2 \leq d \leq r+1$ [and where this upper bound on d is imposed by Singleton's bound (8.20)]. We can choose $\mathbf{c}_1$, where $\mathbf{c}_1^T$ is the first column of H, to be any non-zero vector in $GF(q)^r$. We continue then to pick further columns $\mathbf{c}_2^T, \mathbf{c}_3^T, \ldots$[until we have chosen $\mathbf{c}_N^T$ where N is the desired blocklength] in such a manner that $\mathbf{c}_i$ is not a linear combination of $d-2$ or fewer of

the vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{i-1}$. Suppose that $\mathbf{c}_i$ has just been chosen. Because every linear combination of $d-2$ or fewer of the vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_i$ is either the zero vector or a linear combination of exactly j of these vectors in which all j coefficients are non-zero and $1 \leq j \leq d-2$, it follows that there are at most

$$\binom{i}{0} + \binom{i}{1}(q-1) + \binom{i}{2}(q-1)^2 + \cdots + \binom{i}{d-2}(q-1)^{d-2} = V_{d-2}^{(i)}$$

distinct linear combinations of $d-2$ of fewer of the vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_i$, where we have written $V_{d-2}^{(i)}$ to denote the volume of the Hamming sphere of radius $d-2$ in the i-dimensional vector space $GF(q)^i$. Thus, if $V_{d-2}^{(i)} < q^r$, there will certainly exist a choice for $\mathbf{c}_{i+1}$ that is not a linear combination of $d-2$ or fewer of the vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_i$. We have proved the following result.

Varshamov's Theorem: For any $GF(q)$ and any positive integers N, K and d with $2 \leq d \leq N-K+1$, if

$$V_{d-2}^{(N-1)} < q^{N-K}, \tag{8.22}$$

then there exists a q-ary (N, K) linear code with $d_{\min} \geq d$.

Example 8.10.1 For any $GF(q)$, choose $d = 3$ and $N - K = m$ where $m \geq 2$. By Varshamov's Theorem, there exists a q-ary $(N, K = N - m)$ linear code with $d_{\min} \geq 3$ provided

$$V_1^{(N-1)} = 1 + (N-1)(q-1) < q^m$$

or, equivalently, if $N - 1 < (q^m - 1)/(q-1)$ or, again equivalently, if $N \leq (q^m - 1)/(q-1)$. Thus, there exists a q-ary $(N = (q^m - 1)/(q-1), K = N - m)$ linear code with $d_{\min} \geq 3$. In fact, these are precisely the Hamming single-error correcting codes (see Problems 8.4, 8.5 and 8.6) and their minimum distance is exactly 3.

In Problem 8.2, another general construction of linear codes is given that we have called there "Gilbert's bound" because the construction is so similar to that used earlier by Gilbert for (in general) nonlinear codes. The construction in Problem 8.2 establishes the following result.

A Gilbert-like Theorem: For any $GF(q)$ and any positive integers N, K and d with $1 \leq d \leq N-K+1$, if

$$\frac{1}{q} V_{d-1}^{(N)} < q^{N-K}, \tag{8.23}$$

then there exists a q-ary (N, K) linear code with $d_{\min} \geq d$.

It is easy to see that (if we ignore the trivial case $d = 1$) the Gilbert-like Theorem is weaker than Varshamov's Theorem in the sense that any code guaranteed to exist by the former theorem is also guaranteed to exist by the latter, but not conversely. [The reader can easily check that the Gilbert-like Theorem does not suffice to establish the existence of the Hamming codes.] The reason for this is the inequality

$$V_{d-2}^{(N-1)} < \frac{1}{q} V_{d-1}^{(N)} \tag{8.24}$$

that holds for all positive integers d and N with $2 \leq d \leq N$. To see the truth of (8.24), it suffices to note that every point $\mathbf{x} = [x_1, x_2, \ldots, x_{N-1}]$ in the Hamming sphere of radius $d-2$ centered at $\mathbf{0}$ in $GF(q)^{N-1}$, i.e., every $\mathbf{x}$ in $GF(q)^{N-1}$ with $w(\mathbf{x}) \leq d-2$, can be associated uniquely with the q points $\{[x_1, x_2, \ldots, x_{N-1}, y] : y \in GF(q)\}$ in the sphere of radius $d-1$ centered at $\mathbf{0}$ in $GF(q)^N$, but that there will be points in the latter sphere not associated to any point in the former (namely all the points $[x_1, x_2, \ldots, x_{N-1}, 0]$ with $w(\mathbf{x}) = d-1$).

The weaker bound (8.23) is more convenient for asymptotic arguments than the stronger bound (8.22). If we fix N and K, then, for the largest d such that (8.23) is satisfied, it will be true that

$$\frac{1}{q} V_d^{(N)} \geq q^{N-K},$$

or, equivalently, that

$$V_d^{(N)} \geq q^{N-K+1}. \tag{8.25}$$

Making use of the well-known bound

$$V_d^{(N)} \leq 2^{N\, h(d/N)} \text{ if } d/N \leq 1/2 \tag{8.26}$$

for $q = 2$ where $h(p) = -p \log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function, we obtain the following bound from (8.25).

Asymptotic Varshamov-Gilbert Bound: For any N and any code rate $R = K/N$, there exists a binary (N, K) linear code with

$$h(d_{\min}/N) > 1 - R. \tag{8.27}$$

Example 8.10.2 For $R = 1/2$ and any even N, it follows from (8.27) that there exists an $(N, K = N/2)$ binary linear code with $h(d_{\min}/N) > 1/2$ or, equivalently, with

$$d_{\min} \geq 0.11N. \tag{8.28}$$

For large N, the best "effective" constructions known for $R = 1/2$ codes have $d_{\min}$ far less than the 11% of N guaranteed by the Varshamov-Gilbert bound.

It is presently unknown whether, for very large N, there exist binary codes that can do better than the bound (8.27). More precisely, it is unknown whether there exist binary codes of length N and rate R $(0 < R < 1)$ for arbitrarily large N such that $h(d_{\min}/N) \geq 1 - R - \Delta_R$ where Δ_R is a positive number. This question of the asymptotic tightness of the Varshamov-Gilbert bound for binary codes is one of the most intriguing open questions in coding theory.

One can easily derive an asymptotic form of the Varshamov-Gilbert bound for $q > 2$, but there is no reason to believe that this non-binary asymptotic bound should be tight for large N. [There are "hand-waving" arguments suggesting that the binary Varshamov-Gilbert bound might be tight for large N.] In fact, for $q \geq 49$, recent code constructions based on algebraic geometry give q-ary linear codes for which $d_{\min}/N$ exceeds the asymptotic Varshamov-Gilbert bound for arbitrarily large N.

8.11 The Multiplicative Group of $GF(q)$

The non-zero elements F^* of any field F form an abelian group. In the case of the finite field $GF(q)$, this multiplicative group $GF(q)^*$ has order $q-1$. From the fact that every element of a finite group has finite order and generates a cyclic group with this order and from the fact that the order of a subgroup of a finite group divides the order of the group (Lagrange's Theorem), it follows that every element of $GF(q)^*$ is a root of the polynomial equation $x^{q-1} = 1$ or, equivalently, all $q-1$ non-zero elements of $GF(q)$ are zeroes of the polynomial $x^{q-1} - 1$. But a polynomial of degree d $(d \geq 0)$ with coefficients in a field F can have at most d zeroes in F or in any extension field E of F. Thus, β is a zero of $x^{q-1} - 1$ [or, equivalently, $x - \beta$ is a divisor of $x^{q-1} - 1$] if and only if β is a non-zero element of $GF(q)$.

Factorization of $x^{q-1}-1$ and x^q-x: The polynomial $x^{q-1}-1$ [where the coefficients 1 and -1 are elements of $GF(q)$] factors completely into linear factors as follows:

$$x^{q-1}-1=\prod_{\beta\in GF(q)^*}(x-\beta). \tag{8.29}$$

Similarly,

$$x^q-x=\prod_{\beta\in GF(q)}(x-\beta). \tag{8.30}$$

The factorization (8.30) follows upon multiplication on both sides in (8.29) by $x=x-0$.

Example 8.11.1 The reader is invited to check by direct multiplication on the right that, in $GF(5)$,

$$x^4-1=(x-1)(x-2)(x-3)(x-4).$$

Let n be the maximum (multiplicative) order of the elements of $GF(q)^*$. Because the order of every element of an abelian group divides the order of the element of maximum order whenever this maximum order is finite (see Problem 7.10), it follows that the (multiplicative) order of every element of $GF(q)^*$ divides n. Thus, all $q-1$ elements of $GF(q)^*$ are zeroes of the polynomial x^n-1. But n divides $q-1$ so that certainly $n\leq q-1$. On the other hand, x^n-1 has at least $q-1$ distinct zeroes so that $n\geq q-1$. We must conclude that $n=q-1$. Thus, $GF(q)^*$ contains an element of order $q-1$ and hence is the cyclic group of order $q-1$. Any generator of this cyclic group, i.e., any element of $GF(q)^*$ with (multiplicative) order $q-1$, is called a *primitive element* of $GF(q)$. From our knowledge (see Section 7.7) of the cyclic group of order n, we can immediately make the following conclusions.

Fundamental Properties of the Multiplicative Group of GF(q): The multiplicative group of $GF(q)$ is the cyclic group of order $q-1$. This cyclic group has $\varphi(q-1)$ generators, i.e., there are exactly $\varphi(q-1)$ primitive elements in $GF(q)$. If α is a primitive element of $GF(q)$ and $\beta=\alpha^i$, then β has (multiplicative) order $(q-1)/\gcd(q-1,i)$. In particular, there are elements of (multiplicative) order N in $GF(q)^*$ if and only if N is a positive integer that divides $q-1$.

Example 8.11.2 The elements 2 and 3 are the $\varphi(5-1)=\varphi(4)=2$ primitive elements of $GF(5)$ because

$$2^1=2,2^2=4,2^3=3,2^4=1 \quad \text{and} \quad 3^1=3,3^2=4,3^3=2,3^4=1.$$

The elements 1 and 4 $(=-1)$ have multiplicative orders 1 and 2, respectively.

8.12 Reed-Solomon Codes

We now consider the most important linear codes, both theoretically and practically, that have yet been found, the so-called *Reed-Solomon (RS) codes*.

Let $GF(q)$ be any finite field with $q\geq 3$ elements and let N be a divisor of $q-1$ satisfying $N\geq 2$. (The usual choice is $N=q-1$ but sometimes one is interested in a smaller N.) Let α be an element of (multiplicative) order N in $GF(q)$ [and we note that, as explained in the previous section, such an α

always exists.] Let m_o be any integer satisfying $0 \leq m_o < N$; m_o is a parameter of secondary importance and is often chosen as either 0 or 1. Let d be an integer satisfying $2 \leq d \leq N$. Then the *Reed-Solomon Code RS(q, N, α, m_o, d)* is the q-ary linear code of block length N for which

$$H = \begin{bmatrix} (\alpha^{m_o})^{N-1} & (\alpha^{m_o})^{N-2} & \cdots & \alpha^{m_o} & 1 \\ (\alpha^{m_o+1})^{N-1} & (\alpha^{m_o+1})^{N-2} & \cdots & \alpha^{m_o+1} & 1 \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ (\alpha^{m_o+d-2})^{N-1} & (\alpha^{m_o+d-2})^{N-2} & \cdots & \alpha^{m_o+d-2} & 1 \end{bmatrix} \tag{8.31}$$

is a parity-check matrix.

We now determine the dimension K and the minimum distance $d_{\min}$ of the RS(q, N, α, m_o, d) code. Note that H in (8.31) has $d-1$ rows. If we consider the square submatrix formed by selecting columns $N-i_1, N-i_2, \ldots, N-i_{d-1}$ of H (where $N > i_1 > i_2 > \cdots > i_{d-1} \geq 0$), we see that this matrix has the form

$$\tilde{H} = \begin{bmatrix} (\alpha^{m_o})^{i_1} & (\alpha^{m_o})^{i_2} & \cdots & (\alpha^{m_o})^{i_{d-1}} \\ (\alpha^{m_o+1})^{i_1} & (\alpha^{m_o+1})^{i_2} & \cdots & (\alpha^{m_o+1})^{i_{d-1}} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ (\alpha^{m_o+d-2})^{i_1} & (\alpha^{m_o+d-2})^{i_2} & \cdots & (\alpha^{m_o+d-2})^{i_{d-1}} \end{bmatrix}$$

We now aim to prove that this square matrix is nonsingular, i.e., that it has linearly independent rows (or, equivalently, linearly independent columns). We can factor $(\alpha^{m_o})^{i_j}$ $[\neq 0]$ out of the j-th column of this submatrix without altering whether the columns are linearly independent or linearly dependent. Doing so, we obtain the new submatrix

$$\hat{H} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_{d-1}} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ (\alpha^{i_1})^{d-2} & (\alpha^{i_2})^{d-2} & \cdots & (\alpha^{i_{d-1}})^{d-2} \end{bmatrix}$$

that we now recognize as Vandermonde's matrix of order $d-1$ (see Problem 8.10). Because α has multiplicative order N and $0 \leq i_{d-1} < \cdots < i_2 < i_1 < N$, it follows that the elements $\alpha^{i_1}, \alpha^{i_2}, \ldots, \alpha^{i_{d-1}}$ are all distinct and hence that Vandermonde's matrix is nonsingular (see Problem 8.10). Thus, $d-1$ columns selected from H are always linearly independent. But any d columns selected from H must be linearly dependent since these columns have only $d-1$ components. It follows that $d_{\min} = d$ for the RS code. But our argument above also implies that the rows of H are linearly independent. Thus, the dimension K of our RS code satisfies $N-K = d-1$ or, equivalently, $d_{\min} = N-K+1$, which shows that our RS code is a maximum-distance-separable (MDS) code. We summarize our findings.

Parameters of Reed-Solomon Codes: The RS(q, N, α, m_o, d) code is a q-ary linear (N, K) code with minimum distance $d_{\min} = d$ and is MDS, i.e., $d_{\min} = N-K+1$.

We now wish to find an efficient decoding algorithm for RS codes. For this purpose, we will introduce the discrete Fourier transform (DFT), which will also give us much better insight into the structure of RS codes.

8.13 The Discrete Fourier Transform

With any vector $\mathbf{b} = [b_1, b_2, \ldots, b_N] \in F^N$, we can identify the polynomial

$$b(X) = b_1 X^{N-1} + b_2 X^{N-2} + \cdots + b_{N-1} X + b_N = \sum_{n=1}^{N} b_n X^{N-n} \tag{8.32}$$

of degree less than N with coefficients in the field F and we can talk interchangeably about the vector $\mathbf{b}$ or the polynomial $b(X)$. Suppose that there is an element α of (multiplicative) order N in F. Then the *discrete Fourier transform (DFT)* of the "time-domain" vector $\mathbf{b}$ is the "frequency-domain" vector $\mathbf{B} = [B_1, B_2, \ldots, B_N] \in F^N$, defined by

$$B_i = b(\alpha^i) \quad i = 1, 2, \ldots, N. \tag{8.33}$$

Making use of (8.32), we can write (8.33) explicitly in terms of the components of $\mathbf{b}$ as

$$B_i = \sum_{n=1}^{N} b_n \cdot (\alpha^i)^{N-n}, \tag{8.34}$$

which, because $\alpha^{iN} = 1$, reduces to

$$B_i = \sum_{n=1}^{N} b_n \alpha^{-in}, \quad i = 1, 2, \ldots, N, \tag{8.35}$$

which the reader may find more familiar. [In digital signal processing of sequences (vectors) over the complex field $\mathbb{C}$, one often chooses $\alpha = e^{+j2\pi/N}$ as the element of (multiplicative) order N used to defined the DFT.]

It is easy to see that the DFT is a true "transform", i.e., that it is invertible. For suppose that $\mathbf{b}$ and $\tilde{\mathbf{b}}$ are vectors with the same DFT, i.e., $\mathbf{B} = \tilde{\mathbf{B}}$. It follows then from (8.33) that α^i is a zero of the polynomial $b(X) - \tilde{b}(X)$ for $i = 1, 2, \ldots, N$. But $b(X) - \tilde{b}(X)$ has degree less than N and the zeroes $\alpha^1, \alpha^2, \ldots, \alpha^N$ of this polynomial are all distinct; we must conclude that $b(X) - \tilde{b}(X)$ is the polynomial 0, i.e., $b(X) = \tilde{b}(X)$ or, equivalently, that $\mathbf{b} = \tilde{\mathbf{b}}$. Thus, $\mathbf{b}$ can indeed be recovered from its DFT $\mathbf{B}$.

To see how to recover $\mathbf{b}$ from its DFT $\mathbf{B}$, we consider the polynomial

$$B(X) = B_1 X^{N-1} + B_2 X^{N-2} + \cdots + B_{N-1} X + B_N \tag{8.36}$$

that can be identified with the vector $\mathbf{B}$ and evaluate this polynomial at $\alpha^{-n} = \alpha^{N-n}$. We obtain

$$\begin{aligned} B(\alpha^{-n}) &= \sum_{i=1}^{N} B_i \cdot (\alpha^{-n})^{N-i} \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} b_j \, \alpha^{-ij} \alpha^{in} \end{aligned}$$

or

$$B(\alpha^{-n}) = \sum_{j=1}^{N} b_j \sum_{i=1}^{N} (\alpha^{n-j})^i. \tag{8.37}$$

For $j = n$, the second sum on the right of (8.37) is just the *sum of N 1's in the field*, which we will denote by $((N))$. For $1 \le n \le N$ and $1 \le j \le N$, we have $-N < n - j < N$ and hence $\alpha^{n-j} \ne 1$ if $j \ne n$. But in this case the second sum in (8.37) must be 0 as we now show. Let $\beta = \alpha^{n-j} \ne 1$ and note that $\beta^N = 1$. Then the second sum S in (8.37) becomes

$$S = \beta + \beta^2 + \cdots + \beta^N = \beta(1 + \beta + \cdots + \beta^{N-1});$$

multiplying by $1 - \beta$ gives

$$(1 - \beta)S = \beta(1 - \beta^N) = 0.$$

But $1 - \beta \ne 0$ and hence $S = 0$ as claimed. Thus we have shown that (8.37) reduces to

$$B(a^{-n}) = ((N))b_n.$$

Thus, the *inverse DFT* is given by

$$b_n = \frac{1}{((N))} B(\alpha^{-n}), \quad n = 1, 2, \ldots, N. \tag{8.38}$$

Making use of (8.36), we see that we can write (8.38) explicitly in terms of the components of $\mathbf{B}$ as

$$b_n = \frac{1}{((N))} \sum_{i=1}^{N} B_i \, \alpha^{+ni} \quad n = 1, 2, \ldots, N \tag{8.39}$$

which again may seem more familiar to the reader.

The point of this section is that the DFT can be used in any field. More precisely, if one can find an element of (multiplicative) order N in a field F, then one can define a DFT of length N for that field. It follows from the discussion in Section 8.11 that we can set up a DFT of length N for a finite field $GF(q)$ if and only if N is a divisor of $q - 1$.

We make one final remark before leaving this section. Suppose we use (8.33) to define B_i for all integers $i \ge 0$, not just for $1 \le i \le N$. In this case, we would have $B_{i+N} = b(\alpha^{i+N}) = b(\alpha^i) = B_i$ for all $i \ge 0$, i.e., the frequency-domain sequence $B_0, B_1, B_2, \ldots$ would be *N-periodic* (but its true period might be a divisor of N). Similarly, if we use (8.38) to extend the definition of the time-domain sequence to all $n \ge 0$, we would have $b_{n+N} = \frac{1}{((N))} B(\alpha^{-n-N}) = \frac{1}{((N))} B(\alpha^{-n}) = b_n$ and thus the extended time-domain sequence $b_0, b_1, b_2, \ldots$ would also be N-periodic. We will hereafter often find it convenient to assume that we are dealing with these extended *semi-infinite N-periodic time-domain and frequency-domain sequences*

$$\underset{\sim}{\mathbf{b}} = [b_0, b_1, b_2, \ldots] \tag{8.40}$$

and

$$\underset{\sim}{\mathbf{B}} = [B_0, B_1, B_2, \ldots] \tag{8.41}$$

defined from $\mathbf{b}$ and $\mathbf{B}$ by the relations

$$b_n = b_{n+N} \quad \text{all } n \ge 0, \tag{8.42}$$

$$B_i = B_{i+N} \quad \text{all } i \ge 0. \tag{8.43}$$

Because of (8.42) and (8.43), it follows that we could let all our indices above run from 0 to $N-1$ instead of from 1 to N with no other changes in any of our relations above in this section, and in fact this is what is usually done in digital signal processing.

8.14 Reed-Solomon Codes and the DFT

From the parity-check matrix H of (8.31) that we used to define the RS(q,N,α, m_o, d) code, one sees that

$$\mathbf{b}H^T = [b(\alpha^{m_o}), b(\alpha^{m_o+1}), \ldots, b(\alpha^{m_o+d-2})]$$

where $b(X) = b_1X^{N-1} + b_2X^{N-2} + \cdots + b_{N-1}X + b_N$. From the definition (8.33) of the DFT, it follows that

$$\mathbf{b}H^T = [B_{m_o}, B_{m_o+1}, \ldots, B_{m_o+d-2}]. \tag{8.44}$$

Thus, an *alternative definition of the RS(q,N,α, m_o, d) code* is as the set of time domain sequences $\mathbf{b}$ in $GF(q)^N$ whose DFT vanishes in the band of frequencies from m_o to $m_o + d - 2$ inclusive.

8.15 Linear Feedback Shift Registers and Linear Complexity

A *linear feedback shift register (LFSR)* is a device, as shown in Fig. 8.3, composed of delay cells, constant multipliers and adders. The LFSR is considered initially to be loaded with the digits $s_0, s_1, \ldots, s_{L-1}$ (where L is the *length* of the LFSR), which are elements of some specified field F. The multiplying constants $-c_1, -c_2, \ldots, -c_L$ are also elements of this same field F. The LFSR, started at time instant 0, produces the semi-infinite output sequence

$$\underset{\sim}{\mathbf{s}} = [s_0, s_1, s_2, \ldots] \tag{8.45}$$

of elements of F according to the recursion

$$s_j = -c_1 s_{j-1} - c_2 s_{j-2} - \cdots - c_L s_{j-L} \qquad j = L, L+1, \ldots$$

that we write more conveniently as

$$s_j + c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_L s_{j-L} = 0 \qquad j = L, L+1, \ldots . \tag{8.46}$$

We now seek a convenient algebraic description of this output sequence $\underset{\sim}{\mathbf{s}}$.

We begin by identifying the sequence $\underset{\sim}{\mathbf{s}}$ with the formal power series

$$S(D) = s_0 + s_1 D + s_2 D^2 + \cdots , \tag{8.47}$$

where our choice of D for the indeterminate is made partly to warn the reader that $S(D)$ may not be a polynomial (i.e., it might contain infinitely many non-zero coefficients) and partly because it is convenient to think of D as the *delay operator*.

To describe the LFSR itself, we use first its *connection polynomial* that we define as

$$C(D) = 1 + c_1 D + c_2 D^2 + \cdots + c_L D^L. \tag{8.48}$$

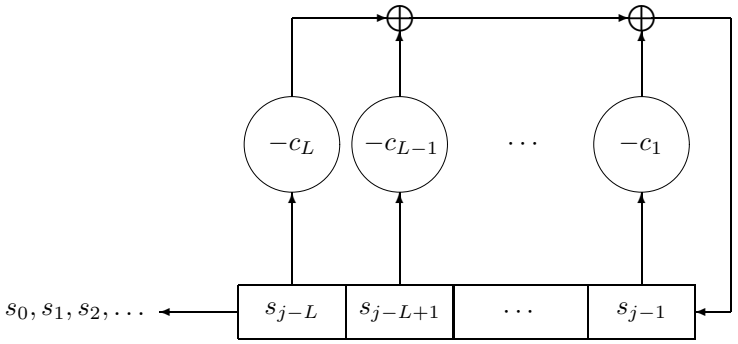

Figure 8.3: A Linear Feedback Shift Register (LFSR)

Note that the degree of $C(D)$ is at most L, but can be smaller. Thus $C(D)$ alone does not suffice to describe the LFSR; we need also to specify the length explicitly. We will write $\langle C(D), L\rangle$ to denote the *LFSR with connection polynomial* $C(D)$ *and length* L.

We now note that the recursion (8.46) specifies that there are no non-zero terms of degree L or greater in the product of $C(D)$ and $S(D)$, i.e., that

$$C(D)S(D) = P(D) \tag{8.49}$$

where $P(D)$ is a polynomial of degree strictly less than L. Writing

$$P(D) = p_0 + p_1 D + \cdots + p_{L-1} D^{L-1} \tag{8.50}$$

and equating terms of degree i, $i < L$, on both sides of (8.49) gives the matrix equation

$$\begin{bmatrix} p_o \\ p_1 \\ \cdot \\ \cdot \\ \cdot \\ p_{L-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ c_1 & 1 & 0 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{L-2} & & \ddots & \ddots & 0 \\ c_{L-1} & c_{L-2} & \cdots & c_1 & 1 \end{bmatrix} \begin{bmatrix} s_o \\ s_1 \\ \cdot \\ \cdot \\ \cdot \\ s_{L-1} \end{bmatrix}$$

which shows that for every choice of $P(D)$ there is a unique corresponding *initial state* $[s_0, s_1, \ldots, s_{L-1}]$ of the LFSR. We summarize our findings.

Description of LFSR Output Sequences: The LFSR $\langle C(D), L\rangle$ can produce the semi-infinite output sequence $\underset{\sim}{\mathbf{s}}$ if and only if the power series $S(D)$ can be written as

$$S(D) = \frac{P(D)}{C(D)} \tag{8.51}$$

where $P(D)$ is a polynomial of degree strictly less than L.

The linear complexity of the semi-infinite sequence $\underset{\sim}{\mathbf{s}}$, which we denote as $L(\underset{\sim}{\mathbf{s}})$, is the smallest L such that $\underset{\sim}{\mathbf{s}}$ can be produced by an LFSR of length L, and is ∞ if no such LFSR exists. By way of convention, the all-zero sequence $\underset{\sim}{\mathbf{0}} = [0, 0, 0, \ldots]$ is said to have linear complexity 0, i.e., $L(\underset{\sim}{\mathbf{0}}) = 0$. It is

also convenient to define linear complexity of finite sequences, say, of $\underset{\sim}{\mathbf{s}}^{(n)} = [s_0, s_1, \ldots, s_{n-1}]$. The linear complexity of $\underset{\sim}{\mathbf{s}}^{(n)}$ is defined as the smallest linear complexity of all semi-infinite sequences having $\underset{\sim}{\mathbf{s}}^{(n)}$ as a prefix. Equivalently, $L(\underset{\sim}{\mathbf{s}}^{(n)})$ is the length of the shortest LFSR $\langle C(D), L\rangle$ that can produce $\underset{\sim}{\mathbf{s}}^{(n)}$ as its first n output digits [where of course the initial state must be the first L digits of $\underset{\sim}{\mathbf{s}}^{(n)}$].

The following is an immediate consequence of our description (8.51) of LFSR output sequences and our definition (8.48) of a connection polynomial $C(D)$ as a polynomial such that $C(0) = 1$.

Linear Complexity of Semi-Infinite Sequences: If the power series $S(D)$ of a semi-infinite sequence $\underset{\sim}{\mathbf{s}}$ can be written as

$$S(D) = \frac{P(D)}{C(D)}$$

where $P(D)$ and $C(D)$ are relatively prime polynomials (i.e., they have no common factor of degree 1 or greater) and $C(0) = 1$, then

$$L(\underset{\sim}{\mathbf{s}}) = \max\{\deg[C(D)], 1 + \deg[P(D)]\}.$$

Moreover, $C(D)$ is the connection polynomial of the unique LFSR of length $L = L(\underset{\sim}{\mathbf{s}})$ that can produce $\underset{\sim}{\mathbf{s}}$.

In practice, one must deal of course only with finite sequences. Thus, the following result is of considerable practical importance.

Linear Complexity of Finite Sequences: If $L(\underset{\sim}{\mathbf{s}}) = L > 0$ and if $\underset{\sim}{\mathbf{s}}^{(n)}$ denotes the first n digits of $\underset{\sim}{\mathbf{s}}$, then

$$L(\underset{\sim}{\mathbf{s}}^{(n)}) = L \quad \text{all } n \geq 2L. \tag{8.52}$$

Moreover, the unique LFSR of length L that can produce $\underset{\sim}{\mathbf{s}}$ is also the unique LFSR of length L that can produce $\underset{\sim}{\mathbf{s}}^{(n)}$ for every $n \geq 2L$.

Proof: Suppose that $\langle C_1(D), L_1\rangle$ and $\langle C_2(D), L_2\rangle$, where $L_1 \leq L$ and $L_2 \leq L$, both produce $\underset{\sim}{\mathbf{s}}^{(n)}$ for some $n \geq 2L$. Equation (8.51) then determines polynomials $P_1(D)$ and $P_2(D)$, with degrees less than L_1 and L_2, respectively, such that

$$\frac{P_1(D)}{C_1(D)} - \frac{P_2(D)}{C_2(D)} = D^n \Delta(D) \tag{8.53}$$

where $\Delta(D)$ is some power series, as follows from the fact that the sequences produced by the two LFSR's have $\underset{\sim}{\mathbf{s}}^{(n)}$ as a prefix and thus can differ at time n at the earliest. Multiplying by $C_1(D)\,C_2(D)$ in (8.53) gives

$$P_1(D)\,C_2(D) - P_2(D)\,C_1(D) = D^n \Delta(D)\,C_1(D)\,C_2(D). \tag{8.54}$$

The left side of (8.54) has degree less than $L_1 + L_2 \leq 2L$ whereas the right side has no terms of degree less than n. But $n \geq 2L$ so that both sides must be 0 and hence

$$\frac{P_1(D)}{C_1(D)} = \frac{P_2(D)}{C_2(D)}.$$

It follows that both LFSR's produce the same semi-infinite sequence. Thus, we have shown that *all* LFSR's of length L or less that produce $\underset{\sim}{\mathbf{s}}^{(n)}$ produce the same semi-infinite sequence, which then must

be the sequence $\underset{\sim}{\mathbf{s}}$ because we know that $\underset{\sim}{\mathbf{s}}$ is produced by some LFSR of length L. Thus, any LFSR of length L or less that produces $\underset{\sim}{\mathbf{s}}^{(n)}$ must be in fact the unique LFSR of length L that produces $\underset{\sim}{\mathbf{s}}$, and hence $L(\underset{\sim}{\mathbf{s}}^{(n)}) = L$. □

There is an efficient algorithm for finding (one of) the shortest LFSR(s) that produces a specified sequence $\underset{\sim}{\mathbf{s}}^{(n)}$ of length n. This *LFSR synthesis algorithm* (or *Berlekamp-Massey algorithm* as it is often called) is given in flowchart form in Fig. 8.4. The interested reader may consult the Appendix of this chapter for a proof that the algorithm does indeed solve the problem of finding (one of) the shortest LFSR(s) that produces $\underset{\sim}{\mathbf{s}}^{(n)}$.

8.16 Blahut's Theorem – Linear Complexity and the DFT

We now show a very interesting connection between linear complexity and the discrete Fourier transform, namely that Hamming weight in one domain equals linear complexity in the other.

Blahut's Theorem: If $\mathbf{B} \in F^N$ is the DFT of $\mathbf{b} \in F^N$ and if $\underset{\sim}{\mathbf{B}}$ and $\underset{\sim}{\mathbf{b}}$ are the corresponding semi-infinite sequences (8.40) and (8.41), then

$$L(\underset{\sim}{\mathbf{B}}) = w(\mathbf{b}) \tag{8.55}$$

and

$$w(\mathbf{B}) = L(\underset{\sim}{\mathbf{b}}). \tag{8.56}$$

Proof: If $\mathbf{b} = \mathbf{0}$, the claim is trivial so we assume $\mathbf{b} \neq \mathbf{0}$ and thus also $\mathbf{B} \neq \mathbf{0}$. Making use of (8.35), we can write the power series for $\underset{\sim}{\mathbf{B}}$ as

$$\begin{aligned} B(D) &= \sum_{i=0}^{\infty} B_i D^i \\ &= \sum_{i=0}^{\infty} \left(\sum_{n=1}^{N} b_n \alpha^{-in} \right) D^i \\ &= \sum_{n=1}^{N} b_n \sum_{i=0}^{\infty} (\alpha^{-n} D)^i \end{aligned}$$

or

$$B(D) = \sum_{n=1}^{N} b_n \frac{1}{1 - \alpha^{-n} D}\,. \tag{8.57}$$

Because α, which defines the DFT, has (multiplicative) order N, it follows that $\alpha^{-1}, \alpha^{-2}, \ldots, \alpha^{-N}$ are all distinct. Thus, the right side of (8.57) is a proper partial fraction with $w(\mathbf{b})$ terms and hence

$$B(D) = \frac{P(D)}{C(D)}$$

where $P(D)$ and $C(D)$ are relatively prime polynomials such that $\deg[P(D)] < \deg[C(D)] = w(\mathbf{b})$ and

$$C(D) = \prod_{n=1 \text{ and } b_n \neq 0}^{N} (1 - \alpha^{-n} D). \tag{8.58}$$

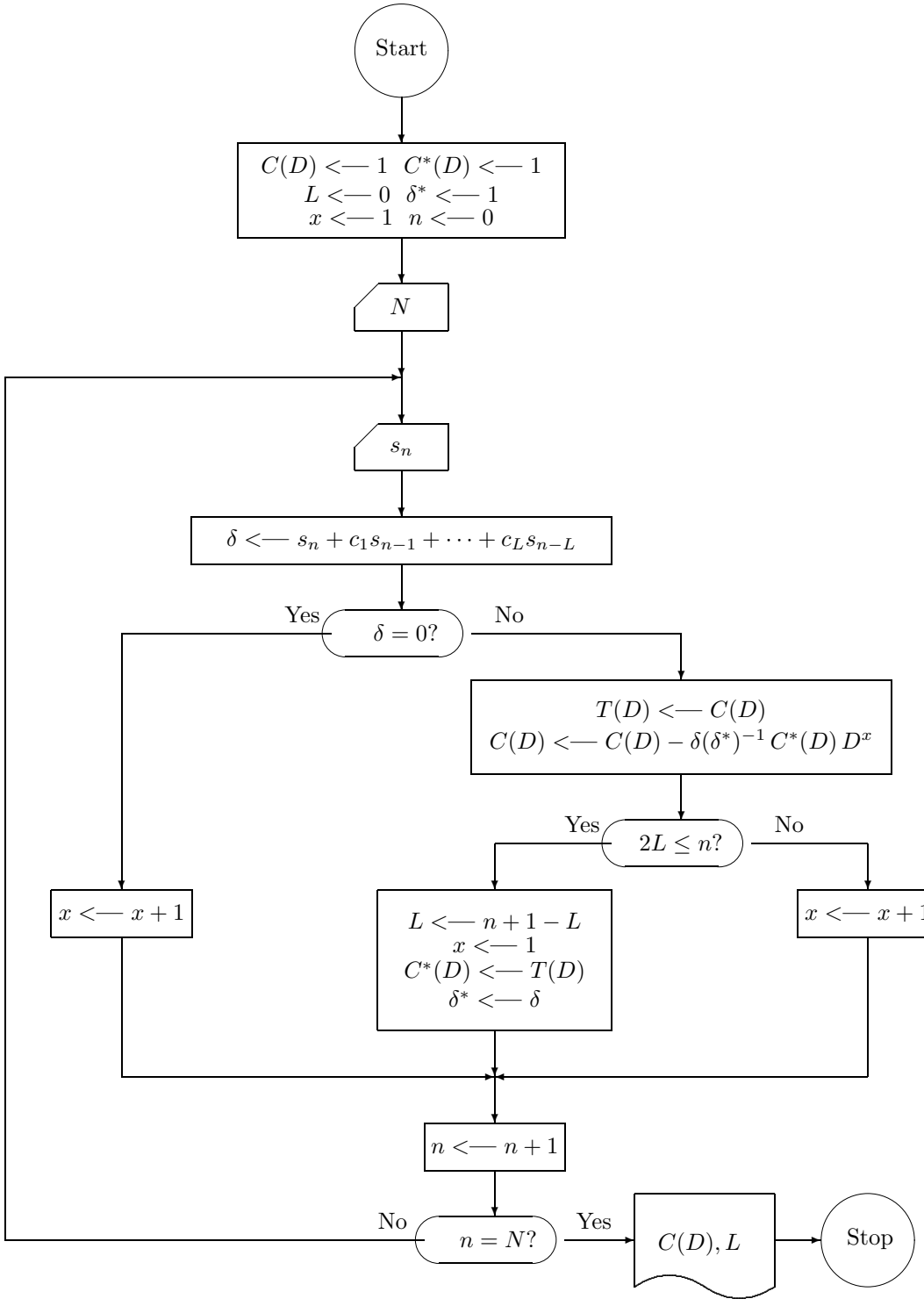

Figure 8.4: LFSR Synthesis Algorithm (Berlekamp-Massey Algorithm) for finding (one of) the shortest LFSR(s) that can generate the sequence $s_0, s_1, \ldots, s_{N-1}$.

It follows from (8.58) that $C(0) = 1$ and thus that $\langle C(D), L = w(\mathbf{b})\rangle$ is the unique shortest LFSR that produces $\underset{\sim}{\mathbf{B}}$. In particular, we have shown that $L(\underset{\sim}{\mathbf{B}}) = \mathrm{w}(\mathbf{b})$. The proof of (8.56) is entirely similar. □

8.17 Decoding the Reed-Solomon Codes

It is now a simple matter to formulate a decoding algorithm for this RS code that will correct all patterns of t or fewer errors where $2t < d = d_{\min}$. Let $\mathbf{b}$ be the transmitted codeword and $\mathbf{e}$ the actual error pattern so that

$$\mathbf{r} = \mathbf{b} + \mathbf{e} \tag{8.59}$$

is the received word. Taking the DFT gives

$$R_i = B_i + E_i \quad \text{all } i.$$

But $B_i = 0$ for $i = m_o, m_o + 1, \ldots, m_o + d - 2$ so that

$$R_i = E_i \qquad m_o \le i \le m_o + d - 2. \tag{8.60}$$

[The reader may notice here that $[R_{m_o}, R_{m_o+1}, \ldots, R_{m_o+d-2}]$ is just the syndrome of $\mathbf{r}$ relative to the parity-check matrix H of (8.31), i.e., computing the syndrome is just the operation of computing $d-1$ terms of the DFT of $\mathbf{r}$.] We see that we now know the first $d-1$ terms of the semi-infinite sequence

$$\underset{\sim}{\mathbf{E}}' = [E_{m_o}, E_{m_o+1}, \ldots, E_{m_o+d-2}, E_{m_o+d-1}, E_{m_o+d}, \ldots]$$

[where the "prime" on $\underset{\sim}{\mathbf{E}}'$ reminds us that this is not quite the same as $\underset{\sim}{\mathbf{E}}$ whose first term is E_o]. But $\underset{\sim}{\mathbf{E}}$ is a periodic sequence and thus $L(\underset{\sim}{\mathbf{E}}') = L(\underset{\sim}{\mathbf{E}})$ since each of these semi-infinite periodic sequences contains the other as a subsequence. Further Blahut's Theorem tells us that $L(\underset{\sim}{\mathbf{E}}) = w(\mathbf{e})$ so that we can now conclude that

$$L(\underset{\sim}{\mathbf{E}}') = w(\mathbf{e}).$$

From our results on the Linear Complexity of Finite Sequences, it follows that if

$$2w(\mathbf{e}) \le d - 1 \tag{8.61}$$

then we can find the shortest LFSR that produces $\underset{\sim}{\mathbf{E}}'$ by applying the LFSR synthesis algorithm to the known first $d-1$ terms of $\underset{\sim}{\mathbf{E}}'$. We can then use this LFSR together with the known terms of $\underset{\sim}{\mathbf{E}}'$ to produce the N-tuple $\mathbf{E}'$ which is just some cyclic shift of $\mathbf{E}$. Knowing $\mathbf{E}$, we can find $\mathbf{e}$ by the inverse DFT and then recover the codeword $\mathbf{b}$ as $\mathbf{r} - \mathbf{e}$. If $2w(\mathbf{e}) \ge d$, however, this algorithm could give incorrect results; we indicate this possiblity by placing "hats" on the computed quantities that are *guaranteed to be correct only when the actual error pattern has Hamming weight satisfying* $2w(\mathbf{e}) < d = d_{min}$. We summarize this efficient decoding algorithm.

Step 1: Compute E_i for $m_o \le i \le m_o + d - 2$ by computing the DFT of the received word $\mathbf{r}$ over this frequency band.

Step 2: Use the LFSR synthesis algorithm to find (one of) the shortest LFSR(s) $\langle C(D), L\rangle$ that produces the finite sequence $E_{m_o}, E_{m_o+1}, \ldots, E_{m_o+d-2}$. If $2L \ge d$, stop and emit **?**, i.e., announce a detected error pattern.

Step 3: Load the LFSR $\langle C(D), L\rangle$ with the last L digits of the sequence $E_{m_o}, E_{m_o+1}, \ldots, E_{m_o+d-2}$ and clock the LFSR $N-(d-1)$ times to produce $\hat{\mathbf{E}}' = [E_{m_o}, E_{m_o+1}, \ldots, E_{m_o+d-2}, \hat{E}_{m_o+d-1}, \ldots, \hat{E}_{m_o+N-1}]$.

Step 4: Cyclically shift $\hat{\mathbf{E}}'$ appropriately (depending on the value m_o) to obtain $\hat{\mathbf{E}} = [\hat{E}_1, \hat{E}_2, \ldots, \hat{E}_N]$.

Step 5: Compute the inverse DFT $\hat{\mathbf{e}}$ of $\hat{\mathbf{E}}$.

Step 6: Subtract $\hat{\mathbf{e}}$ from $\mathbf{r}$ to obtain the decoding decision $\hat{\mathbf{b}}$ for the transmitted codeword.

Chapter 9

AN INTRODUCTION TO CRYPTOGRAPHY

9.1 The Goals of Cryptography

The word "cryptography" evokes in the popular imagination images of spies furtively penning messages in invisible ink, of banal phrases that hide within them some message of vital military or diplomatic importance, of couriers carrying attaché cases handcuffed to their wrists and filled with secret keys, and of mysterious people toiling in a Black Chamber to extract the message hidden deep in long sequences of random-appearing numbers. Indeed, the history of cryptography spans several millenia and is replete with real case histories whose drama equals that of the most imaginative espionage fiction. Only a few decades ago, cryptography was still considered to the province of diplomats and generals. But the sudden dawning of the Information Age in which we now live inevitably meant that cryptography would become another standard tool of the communications engineer. Information can have enormous economic value or can be of such an extremely sensitive nature that its unauthorized disclosure could be highly damaging to some individual. Falsified information can wreak economic damage or can destroy an individual. When such information is transmitted over easily accessible means of communications, prudence demands that its confidentiality and its authenticity be assured. Cryptography can provide that assurance.

As its Greek roots that means "hidden writing" suggest, cryptography is concerned with the secrecy of information; but it is equally concerned with the authenticity of information. The twin goals of cryptography, *secrecy* and *authenticity*, are sometimes not so easy to distinguish. In fact, it is only recently that cryptographers themselves have appreciated that these two goals are in fact quite independent. Xuejia Lai has given perhaps the best rule for distinguishing between secrecy and authenticity. A technique provides *secrecy* if it determines who can *receive* a message; it provides *authenticity* if it determines who can have *sent* a message.

9.2 Shannon's Theory of Perfect Secrecy

In his 1949 paper, *Communication Theory of Secrecy Systems*, Claude Shannon provided the first truly scientific treatement of secrecy. Shannon's theory of secrecy is in fact a straightforward application of the information theory that he had formulated in his celebrated 1948 paper, *A Mathematical Theory of Communication*. The ingenuity of the 1949 paper lies not in the methods used therein but rather in the incisive formulation that Shannon made of the problem of secrecy.

Figure 9.1 shows the general model of a secret-key cryptosystem. This model differs from that used by Shannon in 1949 only by its inclusion of the public random source and the private random source. These randomizing sources have recently been found to be quite useful although, as we shall see, they do not alter the basic results of Shannon's theory.

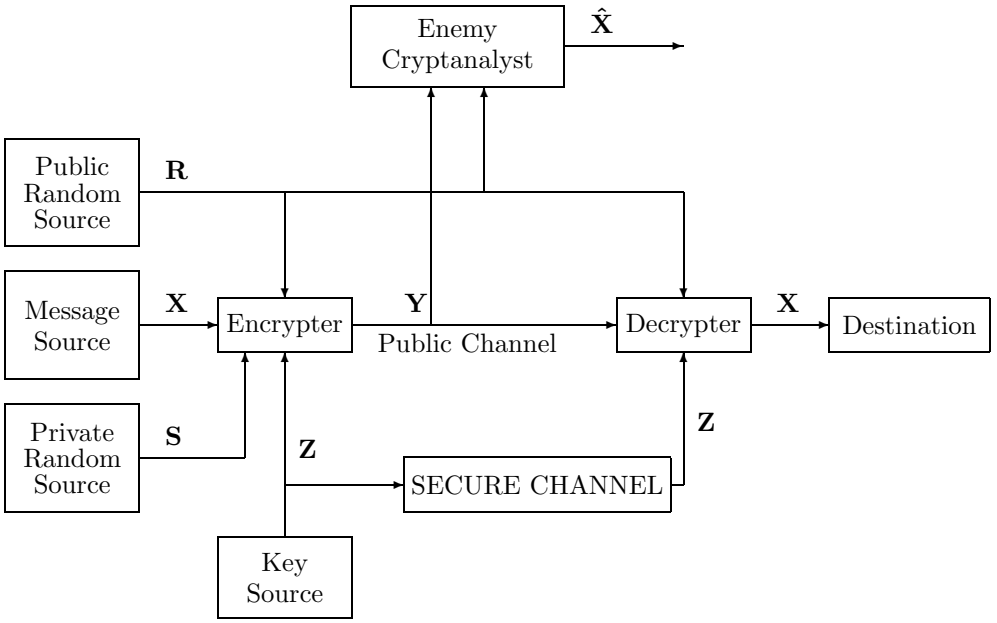

Figure 9.1: General model of a secret-key cryptosystem.

The output of the message source in Fig. 9.1 is the *plaintext* $\mathbf{X} = [X_1, X_2, \ldots, X_M]$ that is to be sent "secretely" to the destination. The key source provides a *secret key* $\mathbf{Z} = [Z_1, Z_2, \ldots, Z_K]$ that is distributed securely to both the encrypter and the decrypter, usually well in advance of the time that the plaintext is generated. This secret key, together with the *private randomizer* $\mathbf{S} = [S_1, S_2, \ldots, S_J]$ and the *public randomizer* $\mathbf{R} = [R_1, R_2, \ldots, R_T]$, determines the particular encrypting transformation used to create the *cryptogram* (or *ciphertext*) $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_N]$. We write

$$\mathbf{Y} = E_{\mathbf{ZRS}}(\mathbf{X}) \tag{9.1}$$

to express the fact that we wish to consider the cryptogram $\mathbf{Y}$ primarily as a function of the plaintext $\mathbf{X}$, the particular function being specified by the values of the secret key $\mathbf{Z}$, the public randomizer $\mathbf{R}$ and the private randomizer $\mathbf{S}$. Note that (9.1) in information-theoretical terms is equivalent to

$$H(\mathbf{Y}|\mathbf{XZRS}) = 0. \tag{9.2}$$

The decrypter has no access to the private randomizer $\mathbf{S}$ but must be able to recover the plaintext $\mathbf{X}$ from the cryptogram $\mathbf{Y}$, the secret key $\mathbf{Z}$ and the public randomizer $\mathbf{R}$. Thus, we must have

$$\mathbf{X} = D_{\mathbf{ZR}}(\mathbf{Y}) \tag{9.3}$$

or, equivalently in information-theoretical terms,

$$H(\mathbf{X}|\mathbf{YZR}) = 0. \tag{9.4}$$

As Fig. 9.1 suggests, the *source outputs* $\mathbf{X}$, $\mathbf{Z}$, $\mathbf{R}$ and $\mathbf{S}$ are all assumed to be statistically independent. A less obvious assumption made by Shannon is that the *secret key is to be used only once* (or, equivalently, that $\mathbf{X}$ is the totality of all plaintexts to be encrypted before the secret key is changed.) Similarly, the public and private randomizers are to be used only once.

It is time now to examine the rascal who makes the cryptosystem necessary, the *enemy cryptanalyst*. [*Cryptanalysis* is the science of "breaking" cryptographic systems.] The enemy cryptanalyst is assumed to observe the cryptogram $\mathbf{Y}$ and the public randomizer $\mathbf{R}$, but nothing else. [This kind of cryptoanalytic attack is called a *ciphertext-only attack*.] His goal is to find the plaintext, at least with some reasonable chance of success.

Shannon made the notion of an "unbreakable cryptosystem" precise by saying that a secret-key cryptosystem provides *perfect secrecy* if the enemy cryptanalyst's observation $(\mathbf{Y}, \mathbf{R})$ is statistically independent of the plaintext $\mathbf{X}$. In information-theoretical terms, perfect secrecy means

$$H(\mathbf{X}|\mathbf{YR}) = H(\mathbf{X}). \tag{9.5}$$

Shannon then went on to derive a quite surprising lower bound on $H(\mathbf{Z})$ for any cryptosystem providing perfect secrecy. Shannon's argument was the soul of simplicity, namely

$$\begin{aligned} H(\mathbf{X}|\mathbf{YR}) &\leq H(\mathbf{XZ}|\mathbf{YR}) \\ &= H(\mathbf{Z}|\mathbf{YR}) + H(\mathbf{X}|\mathbf{YRZ}) \\ &= H(\mathbf{Z}|\mathbf{YR}) \\ &\leq H(\mathbf{Z}) \end{aligned} \tag{9.6}$$

where we have made use of (9.2) and the fact that removing conditioning can only increase uncertainty. The following bound now follows immediately from (9.5) and (9.6).

Shannon's bound on Key Size: In any cryptosystem that provides perfect secrecy, the uncertainty of the secret key must be at least as great as that of the plaintext, i.e.,

$$H(\mathbf{Z}) \geq H(\mathbf{X}). \tag{9.7}$$

If the K digits of the secret key are binary digits, then $K \geq H(\mathbf{Z})$ (in bits) with equality if and only if all 2^K values of the secret keys are equally likely. Thus, Shannon's bound (9.7) states that *there must be at least as many binary digits in the secret key as there are bits of information in the plaintext if the cryptosystem provides perfect secrecy!*

The question whether any cryptosystem can provide perfect secrecy was also answered by Shannon in a way that we now slightly generalize.

Let $\langle G, * \rangle$ be any finite group of order, say, $\#(G) = L$. Suppose that the lengths M, K and N of the plaintext, secret key and cryptogram, respectively, are all the same and that the components of $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{Y}$ are all elements of G. We will call such a cryptosystem a *group-operation cipher* if

$$Y_i = X_i * Z_i, \qquad i = 1, 2, \ldots, N. \tag{9.8}$$

Note that no randomizers are used in such a cipher.

Perfect Secrecy with Group-Operation Ciphers: If the components $Z_1, Z_2, \ldots, Z_N$ of the secret key $\mathbf{Z}$ of a group-operation cipher are statistically independent and equally likely to take on each of the L elements of the group as their value, then the group-operation cipher provides perfect secrecy regardless of the plaintext statistics.

Proof: Note that $\mathbf{Y} = [X_1 * Z_1, X_2 * Z_2, \ldots, X_N * Z_N]$, which we abbreviate by writing $\mathbf{Y} = \mathbf{X} * \mathbf{Z}$. Thus,

$$\begin{aligned} P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) &= P(\mathbf{X} * \mathbf{Z} = \mathbf{y} | \mathbf{X} = \mathbf{x}) \\ &= P(\mathbf{x} * \mathbf{Z} = \mathbf{y} | \mathbf{X} = \mathbf{x}) \\ &= P(\mathbf{Z} = \mathbf{x}^{-1} * \mathbf{y} | \mathbf{X} = \mathbf{x}) \\ &= P(\mathbf{Z} = \mathbf{x}^{-1} * \mathbf{y}) \\ &= L^{-N}, \qquad \text{all } \mathbf{x}, \mathbf{y} \end{aligned} \tag{9.9}$$

where we have used the facts that the secret key $\mathbf{Z}$ and the plaintext $\mathbf{X}$ are statistically independent and that the secret key $\mathbf{Z}$ has equal probability of taking on any of the L^N possible sequences of N group elements as its value. The fact that $P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x})$ does not depend on $\mathbf{x}$ means that $\mathbf{X}$ and $\mathbf{Y}$ are statistically independent, i.e., that the cryptosystem provides perfect secrecy. □

Shannon proved the existence of perfect secrecy systems by showing that the *Vernam cipher* (or *one-time pad* as it is often called) is perfect. The one-time pad is the special case of a group-operation cipher where the group operation is modulo L addition, i.e., $\langle G, * \rangle = \langle \mathbb{Z}_L, \oplus \rangle$, and where the key digits are provided by an L-ary symmetric source. The binary Vernam cipher, which is the most frequently used, is shown in Fig. 9.2.

It is interesting to note that Vernam published his cipher in 1926, but Shannon's 1949 proof of its unbreakability appears to have been the first such proof. [Vernam correctly believed and stated that his cipher was unbreakable, but cryptographic history abounds with inventors of ciphers who incorrectly believed and stated that their ciphers were unbreakable.]

Note that a perfect group-operation cipher uses a secret key whose uncertainty is

$$H(\mathbf{Z}) = N \log_2 L \quad \text{bits},$$

which *meets Shannon's bound (9.7) with equality if and only if the message source is the L-ary symmetric source*, i.e., the L-ary discrete memoryless source with equiprobable output letters. This situation can always be at least approximated by source encoding (data compression) of the actual message source to produce the plaintext. Indeed, as Shannon pointed out in 1949, *source coding (to remove redundancy) before encryption is a powerful way to increase the security of a cryptographic system.*

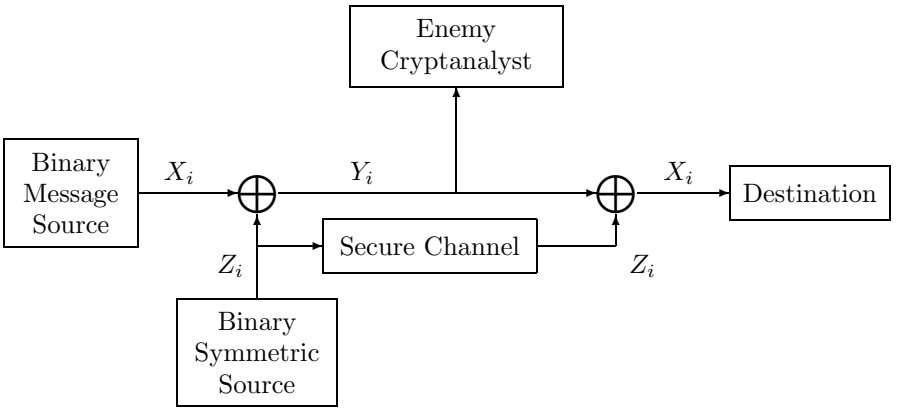

Figure 9.2: The binary Vernam cipher (or "one-time pad").

9.3 Kerckhoffs' Hypothesis

In our discussion of Shannon's theory of perfect secrecy, we tacitly assumed that *everything about the encipherment process* (including the statistics of the plaintext, the secret key, and the public and private randomizers) *is known to the enemy cryptanalyst, except of course for the actual values of the plaintext, the secret key and the private randomizer.* In particular, the enciphering algorithm itself is *not* secret. Thus, any secrecy that the cipher provides owes its existence to the fact that the enemy cryptanalyst does not know the actual value of the secret key. The hypothesis that the security of the cipher should reside entirely in the secret key was first made in 1883 by the Dutchman Auguste Kerckhoffs (1835-1903); cryptographic history has demonstrated its wisdom. A determined enemy is generally able to obtain a complete "blueprint" of the enciphering and deciphering machines, either by clever deduction or by outright stealing or by measures in-between these extremes. It is generally a risky proposition to rely on the hope that one can safeguard the design of one's cipher from such an enemy.

Nonetheless, in many applications of cryptography and notably in military and diplomatic applications, the cryptographer often makes a strenuous attempt to keep his enciphering algorithm secret. Kerckhoffs would not disapprove of this, but would warn such a cryptographer not to count too much on the success of this safekeeping. Kerckhoffs would positively admire, however, the designers of the Data Encryption Standard (DES) who in 1976 published a complete description of their cipher, which is perhaps the most widely used cipher today.

9.4 Imperfect Secrecy and Unicity Distance

To determine when a non-randomized cipher that did not offer perfect secrecy could in principle be broken, Shannon introduced in 1949 the so-called *key equivocation function* $f(\cdot)$ defined for every nonegative integer n by

$$f(n) = H(\mathbf{Z}|Y_1Y_2\cdots Y_n) \tag{9.10}$$

where, by way of convention, $f(0) = H(\mathbf{Z})$. He then defined the *unicity distance* n_u as the smallest n such that $f(n) \approx 0$. Thus, n_u is the least amount of ciphertext from which the enemy cryptanalyst is able to determine the secret key essentially uniquely – at least in principle, finding $\mathbf{Z}$ from $[Y_1, Y_2, \ldots, Y_{n_u}]$ may very well be an intractable problem. One often says, somewhat imprecisely, that the unicity distance is the *least amount of ciphertext for which the enemy cryptanalyst can break the cipher* in a ciphertext-only attack.

For a special type of non-randomized cipher, Shannon showed that the key equivocation function behaves as shown in Fig. 9.3, i.e., a linear decrease with n until $f(n) \approx 0$. The slope of this linear decrease is $-\rho$ where ρ is the *percentage redundancy of the plaintext information as represented in the N-digit cryptogram*, i.e.,

$$\rho = 1 - \frac{H(\mathbf{X})}{N \log L_y} \tag{9.11}$$

where L_y is the size of the ciphertext alphabet and N is the length of the cryptogram.

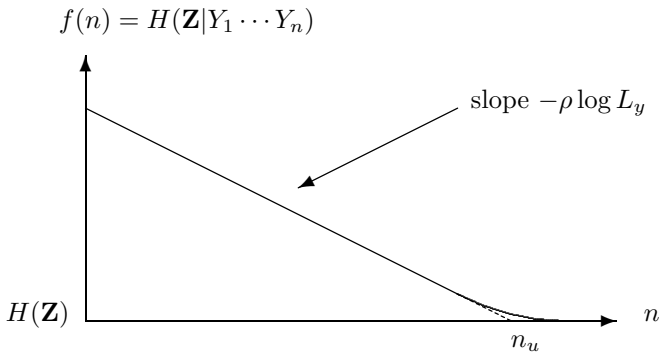

Figure 9.3: The usual form of the key equivocation function for a secret-key cipher.

The corresponding unicity distance is

$$n_u \approx \frac{H(\mathbf{Z})}{\rho \log L_y}. \tag{9.12}$$

We give here an alternative derivation of Shannon's formulas (9.11) and (9.12) that perhaps better explains why they hold for virtually all practical non-randomized ciphers. A design goal of virtually all cipher designers is that the ciphertext appear to be totally random for as long as possible, i.e., that

$$H(Y_1 Y_2 \cdots Y_n) \approx n \log L_y \tag{9.13}$$

holds over as large a range of n as possible. Because the cipher is non-randomized, the uncertainty of $[Y_1, Y_2, \ldots, Y_n]$ when $\mathbf{Z}$ is given will roughly equal the uncertainty of that part of $\mathbf{X}$ that determines $[Y_1, Y_2, \ldots, Y_n]$. For most ciphers and most real message sources, a good approximation will be

$$H(Y_1 Y_2 \cdots Y_n|\mathbf{Z}) \approx \frac{n}{N} H(\mathbf{X}) \tag{9.14}$$

for all n. The two assumptions (9.13) and (9.14) suffice to obtain Shannon's result since

$$\begin{aligned} H(Y_1 Y_2 \cdots Y_n \mathbf{Z}) &= H(\mathbf{Z}) + H(Y_1 Y_2 \cdots Y_n|\mathbf{Z}) \\ &\approx H(\mathbf{Z}) + \frac{n}{N} H(\mathbf{X}) \end{aligned} \tag{9.15}$$

where we have used (9.14), and also

$$\begin{aligned} H(Y_1Y_2\cdots Y_n\mathbf{Z}) &= H(Y_1Y_2\cdots Y_n) + H(\mathbf{Z}|Y_1Y_2\cdots Y_n) \\ &\approx n\log L_y + f(n) \end{aligned}$$

where we have used (9.13) and the definition (9.10). Equating these two approximations yields

$$\begin{aligned} f(n) &\approx H(\mathbf{Z}) - n(\log L_y)\Big[1 - \frac{H(\mathbf{X})}{N\log L_y}\Big] \\ &= H(\mathbf{Z}) - n\rho\log L_y, \end{aligned} \tag{9.16}$$

which is the straight line plotted in Fig. 9.3. Shannon's formulas (9.11) and (9.12) are now immediate. [The reader may well wonder why (9.16) does not also hold for $n > n_u$, which would give the patently false result that $f(n) < 0$ for $n > n_u$. The reason is that

$$\begin{aligned} H(Y_1Y_2\cdots Y_n) &\le H(\mathbf{Z}\,Y_1Y_2\cdots Y_n) \\ &\approx H(\mathbf{Z}) + \frac{n}{N}H(\mathbf{X}) \end{aligned}$$

where we have used (9.15). Thus, the "total randomness" assumption (9.13) for $[Y_1, Y_2, \ldots, Y_n]$ is possible only if

$$n\log L_y \le H(\mathbf{Z}) + \frac{n}{N}H(\mathbf{X})$$

or, equivalently, only if

$$n \le n_u.$$

Thus, our assumption (9.13) is valid just up to the point where we require its validity to derive (9.14), but not for larger n.]

Problem 9.1. considers the unicity distance of a so-called *stream cipher*, where the validity of Shannon's formulas (9.11) and (9.12) are even more apparent.

Shannon was well aware that the formulas (9.11) and (9.12) were generally valid. He remarked explicitely that the analysis that he used for a very particular type of cipher "can be used to estimate equivocation characteristics and the unicity distance for the ordinary types of ciphers". Indeed, cryptographers today routinely use Shannon's formulas to estimate the unicity distance of virtually all ciphers.

It is important to note that Shannon's determination of the unicity distance assumes a ciphertext-only attack by the enemy cryptanalyst, i.e., n_u is the amount of ciphertext required by the cryptanalyst to determine the key essentially uniquely in a ciphertext-only attack. Moreover, Shannon's analysis assumes a non-randomized cipher, but it is not hard to see that this analysis is not affected by the use of a public randomizer in the encryption process.

If a private randomizer is used, however, Shannon's analysis must be altered slightly. The usual purpose of such a randomizer is to remove redundancy from the data actually encrypted by "pumping in" additional randomness in such a way that, after decryption, the legitimate receiver can extract the intended message without knowledge of the private randomizer. For instance, in typical English text (written with the 26 letters of the Latin alphabet and the space symbol), the most likely symbol is a space which has probability about .1859, the next most likely is an "e" which has probability about .1031, etc., and the least likely is a "z" which has probability about .0005. Suppose that such English text is converted into 11-bit symbols in the manner that 381 ($\approx .1859 \times 2048$) of the 2048 such symbols are all used to represent a "space", 211 ($\approx .1031 \times 2048$) such symbols are used to represent an "e", etc.,

and 1 ($\approx .0005 \times 2048$) such symbol is used to represent a "z", the choice of a substitute for a letter in the English text being made by a uniform random choice (determined by the secret randomizer) from the set of 11-bit substitutes for that letter. The substitutes for successive digits of the plaintext are chosen independently; the digits in the private randomizer are used to make the required random choices. After such conversion, each 11-bit symbol in the converted text is essentially equally likely to take on any of the 2048 possible values. Moreover, anyone who sees the converted text can return it to its original form simply by replacing each of the 381 substitutes for a space with a "space", replacing each of the 211 substitutes for an e with an "e", etc.. There is no need to know in advance which substitutes were randomly chosen, but the receiver in fact learns what choices were made in the process of reconverting the modified text to its original form. This is the process of *homophonic substitution* (several different "homophones" represent each letter of the original alphabet) and it is an old cryptographic trick for increasing the unicity distance of a cipher. If the "plaintext" $\tilde{\mathbf{X}}$ for a conventional non-randomized cipher is in fact the result of homophonic coding of the true plaintext $\mathbf{X}$ with the aid of the private randomizer $\mathbf{S}$, then (because the legitimate receiver obtains both $\mathbf{X}$ and $\mathbf{S}$ when he reconverts the "plaintext" $\tilde{\mathbf{X}}$) all the results of this section still apply provided that $H(\mathbf{X})$ is replaced by $H(\tilde{\mathbf{X}}) = H(\mathbf{X}) + H(\mathbf{S})$.

Finally, the results of this section can be applied to the case where the enemy cryptanalyst is able to make a *known-plaintext attack* (in which the enemy cryptanalyst knows the plaintext $\mathbf{X}$) or a *chosen-plaintext attack* (in which the enemy has chosen $\mathbf{X}$ himself) in his effort to find the secret key $\mathbf{Z}$. One simply replaces $H(\mathbf{X})$ by 0 so that the appropriate redundancy is $\rho = 1$.

9.5 Simmons' Theory of Authenticity

Although Shannon gave his theory of secrecy in 1949, it was not until 1984 that Simmons gave an analogous theory of authenticity.

Simmons considered a secret-key algorithm as in Fig. 9.1 with a one-time key, except that he did not allow a public random source. Until further notice, we shall likewise assume that there is no public randomizer $\mathbf{R}$.

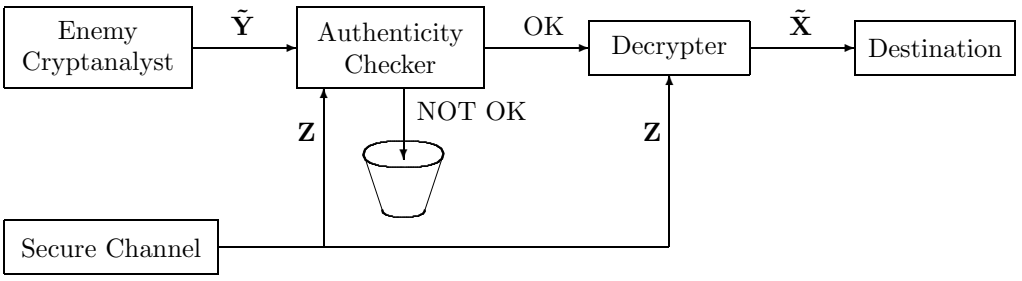

Figure 9.4: Model of an impersonation attack

Simmons considered various types of authenticity attacks, the simplest and most fundamental of which is the *impersonation attack* shown in Fig. 9.4. In this attack, the enemy cryptanalyst tries to

pawn his "crooked cryptogram" $\tilde{\mathbf{Y}}$ off on the destination as the real thing. The attack succeeds if $\tilde{\mathbf{Y}}$ is accepted as authentic, i.e., if $\tilde{\mathbf{Y}}$ is in fact a valid cryptogram for the actual secret key $\mathbf{Z}$. The purpose of such an impersonation attack might, for instance, be to disrupt legitimate communications. It is important to note that the enemy cryptanalyst makes no observation of the actual cryptogram $\mathbf{Y}$, which in fact may not yet exist. It is even possible (but improbable) that he will choose $\tilde{\mathbf{Y}}= \mathbf{Y}$, but this is still counted as a win for the enemy cryptanalyst in an impersonation attack.

To describe the "authenticity checker" in Fig. 9.4, we need to introduce the *authentication function* $\varphi(\cdot,\cdot)$, which is defined by

$$\varphi(\mathbf{y},\mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{y} \text{ can be a valid cryptogram when } \mathbf{z} \text{ is the secret key} \\ 0 & \text{otherwise.} \end{cases}$$

The authenticity checker is merely a device that accepts the crooked cryptogram $\tilde{\mathbf{Y}}$ if $\varphi(\tilde{\mathbf{Y}},\mathbf{Z}) = 1$ and rejects it otherwise.

Simmons has defined the *impersonation success probability* P_I as the probability that $\tilde{\mathbf{Y}}$ is accepted as valid, given that the enemy cryptanalyst uses the attack that is optimum in the sense of maximizing this probability. Let $A_{\mathbf{y}}$ denote the *event that* $\mathbf{y}$ *will be accepted by the decrypter as a valid cryptogram.* The probability of $A_{\mathbf{y}}$ is then

$$P(A_{\mathbf{y}}) = \sum_{\mathbf{z}} \varphi(\mathbf{y},\mathbf{z}) P_{\mathbf{Z}}(\mathbf{z}), \tag{9.17}$$

i.e., the total probability of all keys for which $\mathbf{y}$ can be a valid cryptogram. The optimum impersonation attack is then to choose $\tilde{\mathbf{Y}}$ as (one of) the value(s) $\mathbf{y}$ that maximizes $P(A_{\mathbf{y}})$. Thus,

$$\begin{aligned} P_I &= \max_{\mathbf{y}} P(A_{\mathbf{y}}) \\ &\geq \sum_{\mathbf{y}} P(A_{\mathbf{y}}) P_{\mathbf{Y}}(\mathbf{y}), \end{aligned} \tag{9.18}$$

where equality holds if and only if $P(A_{\mathbf{y}})$ is the same for all crytpograms $\mathbf{y}$ having positive probability, i.e., if and only if all authenticity attacks are optimum (or, equivalently, all are equally bad). Using (9.17) in (9.18) gives

$$P_I \geq \sum_{(\mathbf{y},\mathbf{z})} \varphi(\mathbf{y},\mathbf{z}) P_{\mathbf{Y}}(\mathbf{y}) P_{\mathbf{Z}}(\mathbf{z}). \tag{9.19}$$

Recall that the *support* of a function is the set of arguments for which the function is non-zero. For instance, the support of the joint probability distribution $P_{\mathbf{YZ}}$, denoted supp $P_{\mathbf{YZ}}$, is the set of all $(\mathbf{y},\mathbf{z})$ such that $P_{\mathbf{YZ}}(\mathbf{y},\mathbf{z}) \neq 0$. But $\varphi(\mathbf{y},\mathbf{z})P_{\mathbf{Z}}(\mathbf{z}) \neq 0$ precisely when $(\mathbf{y},\mathbf{z}) \in$ supp $P_{\mathbf{YZ}}$. Thus, (9.19) can be rewritten as

$$\begin{aligned} P_I &\geq \sum_{(\mathbf{y},\mathbf{z})\in \text{supp} P_{\mathbf{YZ}}} P_{\mathbf{Y}}(\mathbf{y}) P_{\mathbf{Z}}(\mathbf{z}) \\ &= \sum_{(\mathbf{y},\mathbf{z})\in \text{supp} P_{\mathbf{YZ}}} P_{\mathbf{YZ}}(\mathbf{y},\mathbf{z}) \frac{P_{\mathbf{Y}}(\mathbf{y}) P_{\mathbf{Z}}(\mathbf{z})}{P_{\mathbf{YZ}}(\mathbf{y},\mathbf{z})} \\ &= E\left[\frac{P_{\mathbf{Y}}(\mathbf{Y}) P_{\mathbf{Z}}(\mathbf{Z})}{P_{\mathbf{YZ}}(\mathbf{Y},\mathbf{Z})}\right], \end{aligned} \tag{9.20}$$

where we have used the fact that the expectation of any function $f(\mathbf{Y},\mathbf{Z})$ of the discrete random variables $\mathbf{Y}$ and $\mathbf{Z}$ that takes values on the extended real line (which includes $-\infty$ and $+\infty$) is defined by

$$E[f(\mathbf{Y},\mathbf{Z})] = \sum_{(\mathbf{y},\mathbf{z})\in \text{supp} P_{\mathbf{YZ}}} P_{\mathbf{YZ}}(\mathbf{y},\mathbf{z}) f(\mathbf{y},\mathbf{z}).$$

Inequality (9.20) is equivalent to

$$\log P_I \geq \log E\left[\frac{P_{\mathbf{Y}}(\mathbf{Y})P_{\mathbf{Z}}(\mathbf{Z})}{P_{\mathbf{YZ}}(\mathbf{Y},\mathbf{Z})}\right]. \tag{9.21}$$

Because $\log(\dot{})$ is strictly convex-$\cap$ on the positive real line, it follows from (9.21) and Jensen's inequality that

$$\begin{aligned}\log E\left[\frac{P_{\mathbf{Y}}(\mathbf{Y})P_{\mathbf{Z}}(\mathbf{Z})}{P_{\mathbf{YZ}}(\mathbf{Y},\mathbf{Z})}\right] &\geq E\left[\log\frac{P_{\mathbf{Y}}(\mathbf{Y})P_{\mathbf{Z}}(\mathbf{Z})}{P_{\mathbf{YZ}}(\mathbf{Y},\mathbf{Z})}\right] \\ &= -I(\mathbf{Y};\mathbf{Z})\end{aligned} \tag{9.22}$$

with equality if and only if $\frac{P_{\mathbf{Y}}(\mathbf{y})P_{\mathbf{Z}}(\mathbf{z})}{P_{\mathbf{YZ}}(\mathbf{y},\mathbf{z})}$ has the same value for all $(\mathbf{y},\mathbf{z})$ in supp $P_{\mathbf{YZ}}$. Combining (9.20) and (9.21) gives the following fundamental result.

Simmons' Bound on Impersonation Success Probability: The probability of success, P_I, in an optimum impersonation attack satisfies

$$P_I \geq 2^{-I(\mathbf{Y};\mathbf{Z})} \tag{9.23}$$

with equality if and only if both (1) all impersonation attacks (in which $\tilde{\mathbf{Y}}$ is chosen as a cryptogram $\mathbf{y}$ with $P_{\mathbf{Y}}(\mathbf{y}) \neq 0$) are equally good, and (2) $\frac{P_{\mathbf{Y}}(\mathbf{y})P_{\mathbf{Z}}(\mathbf{z})}{P_{\mathbf{YZ}}(\mathbf{y},\mathbf{z})}$ has the same value for all $(\mathbf{y},\mathbf{z})$ in supp $P_{\mathbf{YZ}}$.

From Simmons' bound (9.23), we can draw the at-first surprising conclusion that to *reduce the enemy's chance of success in an impersonation attack to an acceptably small value, one must allow the cryptogram* $\mathbf{Y}$ *to give away much information about the secret-key* $\mathbf{Z}$. In general, however, it may be very difficult for the enemy to exploit this information.

We now consider the modifications to the above analysis that must be made when the secret-key cryptosystem includes a public randomizer $\mathbf{R}$ (in addition to the private randomizer $\mathbf{S}$ that was allowed above). In this case, we assume that the enemy cryptanalyst observes $\mathbf{R}$, but nothing else, before producing his crooked cryptogram $\tilde{\mathbf{Y}}$ in an impersonation attack. The enemy cryptanalyst can thus optimize his attack for the observed value $\mathbf{r}$ of $\mathbf{R}$. Letting $P_{I|\mathbf{r}}$ denote his optimum success probability when the event $\mathbf{R} = \mathbf{r}$ occurs, we see that all the analysis for the case when there was no public randomizer still applies, except that all the probability distributions used must be conditioned on the occurrence of this event. In particular, Simmons' bound (9.23) becomes

$$P_{I|\mathbf{r}} \geq 2^{-I(\mathbf{Y};\mathbf{Z}|\mathbf{R}=\mathbf{r})} \tag{9.24}$$

where $I(\mathbf{Y};\mathbf{Z}|\mathbf{R}=\mathbf{r})$ is the mutual information between the cryptogram $\mathbf{Y}$ and the secret key $\mathbf{Z}$, given that the event $\mathbf{R} = \mathbf{r}$ has occurred. The optimum probability of successful impersonation P_I can be written according to the "theorem of total probability" as

$$P_I = \sum_{\mathbf{r}} P_{I|\mathbf{r}} P_{\mathbf{R}}(\mathbf{r}). \tag{9.25}$$

Using (9.25) in (9.24) now gives

$$P_I \geq \sum_{\mathbf{r}} 2^{-I(\mathbf{Y};\mathbf{Z}|\mathbf{R}=\mathbf{r})} P_{\mathbf{R}}(\mathbf{r}). \tag{9.26}$$

But $2^{-(\cdot)}$ is a strictly convex-$\cup$ function on the whole real line so that Jensen's inequality can be used in (9.26) to give

$$P_I \geq 2^{-\sum_{\mathbf{r}} I(\mathbf{Y};\mathbf{Z}|\mathbf{R}=\mathbf{r})P_{\mathbf{R}}(\mathbf{r})}, \tag{9.27}$$

which is precisely the same as

$$P_I \geq 2^{-I(\mathbf{Y};\mathbf{Z}|\mathbf{R})}.$$

We leave to the reader the task of checking that the conditions for equality in (9.27) are as stated in the following summary.

Extended Simmons' Bound on Impersonation Success Probability: When the secret-key cryptosystem includes a public randomizer $\mathbf{R}$ (as well as possibly also a private randomizer $\mathbf{S}$), then the probability of success, P_I, in an optimum impersonation attack satisfies

$$P_I \geq 2^{-I(\mathbf{Y};\mathbf{Z}|\mathbf{R})} \tag{9.28}$$

with equality if and only if the following three conditions are satisfied:

1. $I(\mathbf{Y};\mathbf{Z}|\mathbf{R}=\mathbf{r})$ has the same value for all $\mathbf{r}$ with $P_{\mathbf{R}}(\mathbf{r}) \neq 0$;
2. all impersonation attacks (in which, when $\mathbf{R}=\mathbf{r}$ is observed, $\tilde{\mathbf{Y}}$ is chosen as a cryptogram $\mathbf{y}$ with $P_{\mathbf{Y}|\mathbf{R}}(\mathbf{y}|\mathbf{r}) \neq 0$) are equally good, and
3. for every $\mathbf{r}$ such that $P_{\mathbf{R}}(\mathbf{r}) \neq 0$, $\dfrac{P_{\mathbf{Y}|\mathbf{R}}(\mathbf{y}|\mathbf{r})P_{\mathbf{Z}}(\mathbf{z})}{P_{\mathbf{YZ}|\mathbf{R}}(\mathbf{y},\mathbf{z}|\mathbf{r})}$ has the same value for all $(\mathbf{y},\mathbf{z})$ in supp $P_{\mathbf{YZ}|\mathbf{R}}$.

This is as much as we will say about Simmons' theory of authenticity, but it should be enough to make clear to the reader that the theory of authenticity is much more subtle than the theory of secrecy. In practice, too, it is usually much harder to ensure authenticity than to ensure secrecy.

9.6 Unconditional Security and Computational Security

To this point we have assumed no computational constraint on the enemy cryptanalyst. Our analysis of techniques for achieving secrecy and/or authenticity applies even when this enemy cryptanalyst has *unlimited time and computing power* available to use in his attack. Security against such a computationally unrestrained enemy is now ususally called *unconditional security*; Shannon used the terminology "theoretical" security. As we have seen, achieving unconditional security usually requires the advance distribution of enormous quantities of secret key, more than is practically acceptable in all but a few esoteric applications of cryptography.

Most cryptographic systems used in practice rely not on the *impossibility* of their being broken but rather on the *difficulty* of such breaking. Security against an enemy who has a certain limited time and computing power available for his attack is now usually called *computational security*; Shannon used the terminology "practical" security.

Shannon postulated in 1949 that every secret-key cipher can in principle be described by a *work characteristic* $W(n)$ that measures its computational security against a ciphertext-only attack. Shannon defined $W(n)$ to be the average computational work (measured in some convenient units, e.g., in hours of computing time on a specific supercomputer) that the enemy would expend to find all values of the secret key consistent with the first n digits of the ciphertext, *given that this enemy used the best possible attacking algorithm.* Shannon further hypothesized that the work function of a typical cipher would have the general shape shown in Fig. 9.5. For $n = 0$, all values of the secret key are consistent with the "empty observation" of the enemy cryptanalyst; no computational work is required to say this. For

small n, it generally becomes more difficult as n increases to say what values of the secret key are no longer possible. But one can expect that, when the number of possible values of the secret key is not too large, it becomes easier as n increases to specify this small set of possible values consistent with the n observed digits of the ciphertext.

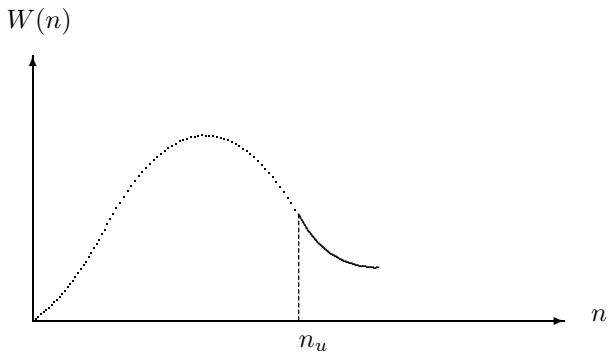

Figure 9.5: Shannon's postulated work characteristic for a typical secret-key cipher (n_u = unicity distance; the dashed portion of the curve indicates that the value of the secret key is not yet uniquely determined).

The fact is, however, that there is no really practical secret-key cipher whose work characteristic is known. It is at the present time not possible to identify the best possible attack on such a cipher, nor is it even possible to give useful lower bounds on the computational effort of the best possible attack. Instead, cryptographers today must rely on what one might call the *historical work characteristic* $W_h(n)$ of the cipher, which we can define as the average computational work that the enemy would expend to find all values of the secret key consistent with the first n digits of the ciphertext, *given that the enemy used the best possible attacking algorithm known to the cryptographer.* How good such a measure is depends obviously on the skill of the cryptographer.

The design of secret-key ciphers is today far more art than science. Such designs are generally guided by two principles suggested by Shannon: confusion and diffusion. The enciphering transformations should greatly complicate the manner in which the statistics of the plaintext affect the statistics of the ciphertext, i.e., they should create *confusion.* To forestall divide-and-conquer attacks, a single plaintext digit and/or a single secret-key digit should influence the values of many ciphertext digits, i.e., the cipher should cause *diffusion* of the plaintext statistics and/or the secret-key statistics. But it is primarily through study of how earlier cryptographic systems have been broken that the cryptographer today develops secret-key ciphers that are (presumably) computationally secure. This is obviously not a very satisfactory state of affairs, but then cryptography is still a young science, albeit an old art.

9.7 Public-key Cryptography

In 1976, two Stanford University researchers, Whitfield Diffie and Martin E. Hellman, published a paper entitled "New Directions in Cryptography", which shook the foundations of cryptographic thinking. Diffie and Hellman suggested that *it is possible to devise computationally secure cryptographic systems*

that required no secure channel for the exchange of secret keys. Ralph Merkle, then a graduate student at the University of California, Berkeley, independently formulated the basic ideas of such "public-key cryptography" and submitted a paper thereon at almost the same time as Diffie and Hellman, but his paper was published almost two years later than theirs so that he has not always received due credit for his discovery.

The Diffie-Hellman paper created intense excitement among cryptographic researchers and, indeed, stimulated a much increased interest in cryptography. It is hardly an exaggeration to say that the Diffie-Hellman paper marks the beginning of active open research in cryptography.

The fundamental contribution of the Diffie-Hellman paper consisted in two crucial definitions, that of a one-way function (which they borrowed from R. M. Needham's work on computer passwords) and that of a trapdoor one-way function (which was completely new), together with suggestions as to how such functions could eliminate the need for the exchange of secret keys in computationally-secure cryptographic systems. Public-key systems can never provide unconditional security; indeed, Shannon and Simmons have given us tight lower bounds on the amount of secret-key that is needed to obtain unconditional security for a secrecy system and for an authenticity system, respectively. The breakthrough provided by Diffie and Hellman was their realization that, if we decide to rely on computational security as indeed we must in most practical cryptographic applications, then we no longer require the secure exchange of secret keys.

9.8 One-Way Functions

A *one-way function* as defined by Diffie and Hellman is a function f such that (1) for every x in the domain of f it is easy to compute $f(x)$, but (2) for virtually all y in the range of f it is computationally infeasible to find an x such that $f(x) = y$. The first thing to note about this definition is that it is only semi-mathematical. The words "easy", "virtually all" and "infeasible" that appear therein are not precise terms, but it is not hard to imagine that they could be given precise meanings should this ever become necessary. The virtue of not precisely specifying these terms is that it does not force us to settle now upon a precise definition that we might later find to be inadequate or to be far inferior to another precise definition; the intuitive meaning of the Diffie-Hellman definition is clear enough to serve our purposes.

It is less clear how a one-way function could be cryptographically useful. The original use of one-way functions in password systems is perhaps the most obvious application. Suppose user i uses the password x_i and presents this password initially to the system, which then computes $y_i = f(x_i)$ and stores the pair (i, y_i) in the password file. When later someone purporting to be user i presents a password $\tilde{x}$ to the system, the system again computes $\tilde{y} = f(\tilde{x})$ and checks whether $(i, \tilde{y})$ is in the password file. If so, the user is granted access; if not, he is denied access to the system. The virtue of this system is that *the password file need not be kept secret* (although there is also no reason to make it public.) An intruder who somehow gets access to the password file obtains no benefit therefrom if f is truly a one-way function. For instance, the entry (i, y_i) is useless to such an intruder as it is computationally infeasible for the intruder to find an x such that $y_i = f(x)$, and he needs such an x to impersonate user i in the access-control protocol. It is interesting to note that this first application of one-way functions was to a system for providing authenticity. It was left to Diffie and Hellman to suggest how certain one-way functions could be used to provide secrecy.

9.9 Discrete Exponentiation and the Diffie-Hellman-Pohlig (Conjectured) One-Way Function

Let $\langle G, * \rangle$ be a cyclic group of order n (see Section 7.7) and let α be a generator of this group. Then by *discrete exponentiation to the base* α in $\langle G, * \rangle$ we mean the function $f : \mathbb{Z}_n \rightarrow G$ such that

$$f(x) = \alpha^x. \tag{9.29}$$

We know from our discussion in Section 7.7 that the n values of $f(x)$ as x ranges over $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ are all distinct. Thus, the inverse function $f^{-1} : G \rightarrow \mathbb{Z}_n$ exists and is called the *discrete logarithm to the base* α. Instead of $x = f^{-1}(y)$, one usually writes the more suggestive

$$x = \log_\alpha y. \tag{9.30}$$

We noted already in Section 7.7 that $\alpha^{x_1} * \alpha^{x_2} = \alpha^{x_1 \oplus x_2}$ where the addition in the exponent is modulo n addition. Equivalently, we can write

$$\log_\alpha (y_1 * y_2) = \log_\alpha y_1 \oplus \log_\alpha y_2. \tag{9.31}$$

Similarly, because for any integer i and x the product ix can be written as $Qn + (i \odot x)$ where $\odot$ denotes multiplication modulo n and Q is some integer, it follows that

$$(\alpha^x)^i = \alpha^{ix} = \alpha^{Qn + (i \odot x)} = (\alpha^n)^Q \alpha^{i \odot x} = \alpha^{i \odot x}.$$

Equivalently, we can write

$$\log_\alpha (y^i) = i \odot \log_\alpha y. \tag{9.32}$$

We see from (9.31) and (9.32) that the discrete logarithm obeys the two usual rules for ordinary logarithms except that *the arithmetic required for the discrete logarithm is that of the ring of integers modulo* n, $\langle \mathbb{Z}_n, \oplus, \odot \rangle$.

One might suspect that, for some choice of the cyclic group $\langle G, * \rangle$ of order n and the base α, the discrete logarithm might be a one-way function because of the following two facts: (1) discrete exponentiation (i.e., given x, computing α^x) is always "easy", requiring at most $2 \log_2 n$ multiplications in G when the square-and-multiply algorithm is used (see Problem 9.4), but (2) the fastest known algorithm (Shank's algorithm) for finding discrete logarithms (i.e., given y, computing $\log_\alpha y$) requires about $\sqrt{n}$ multiplications (see Problem 9.7). However, one must not be too hasty in concluding that discrete exponentiation is one-way. If n has a special form or if the group $\langle G, * \rangle$ has a special structure, there may be major shortcuts to Shank's algorithm; moreover, there is no proof that there are not algorithms for computing the general discrete logarithm that are much faster than Shank's algorithm.

The *Diffie-Hellman-Pohlig conjecture* is that discrete exponentiation in the multiplicative group of $GF(p)$ when the base α is a primitive element of $GF(p)$ is a one-way function provided that p is a large number (say, at least 100 decimal digits in length) such that $n = p - 1$ also has a large prime factor (ideally, $n = p - 1 = 2p'$ where p' is also a prime). [This conjecture was made by Diffie and Hellman in 1976 whithout the proviso that n must have a large prime factor; the necessity of this condition was added by S. C. Pohlig and Hellman in 1978.] The conjecture still stands as of this writing in 1991. No proof of the conjecture has been forthcoming, but neither has an algorithm been found for inverting the

Diffie-Hellman-Pohlig function that is substantially faster than Shank's algorithm. In other words, the historical evidence in favor of the conjecture has been increasing, but hard theoretical evidence has not been produced.

9.10 The Diffie-Hellman Public Key-Distribution System

In their 1976 paper, Diffie and Hellman suggested an ingenious scheme for creating a common secret key between sender and destination in a network without the need for a secure channel to exchange secret keys; their scheme relies on the one-wayness of discrete exponentiation. Suppose $f(x) = \alpha^x$ is truly a one-way function and is known to all users of the network. Diffie and Hellman postulated a *Trusted Public Directory* (TPD) containing non-secret but authentic information that is available to all users of the network.

Upon joining the network, each user (say, user i) secretly and randomly chooses a *private number* x_i and then computes his *public number* $y_i = \alpha^{x_i}$. He identifies himself in some satisfactory way to the custodian of the TPD, who then places the pair (i, y_i) in the TPD. Any user in the network is permitted to fetch any information in this TPD.

When two users, say users i and j, later wish to communicate secretly, each fetches the public number of the other from the TPD, then raises the fetched number to a power equal to his own private number (by square-and-multiply). Thus, user i computes $y_j^{x_i} = (\alpha^{x_j})^{x_i} = \alpha^{x_j x_i}$ and user j computes $y_i^{x_j} = (\alpha^{x_i})^{x_j} = \alpha^{x_i x_j}$. This number $\alpha^{x_i x_j}$ that both user i and user j can compute is their *common secret*, which they can now use as the secret key in a conventional secret-key cryptosystem. What the Diffie-Hellman scheme provides is thus a *public way to distribute secret keys*, and this scheme is usually called the *Diffie-Hellman public key-distribution system.*

An attacker, who wishes to eavesdrop on the secret conversation between user i and user j and who is able to take discrete logarithms, can easily break the system by fetching both y_i and y_j from the TPD, taking the logarithm of one of these (say, $x_i = \log_\alpha y_i$) and then exponentiating by square-and-multiply to get $y_j^{x_i} = \alpha^{x_i x_j}$, the desired secret key. But if the discrete exponentiation used is truly one-way, this attack is computationally infeasible. As of this writing in 1991, no one has produced an attack on the Diffie-Hellman public key-distribution scheme that is not computationally equivalent to computing the discrete logarithm, nor has anyone proved that all attacks on this system are computationally equivalent to computing the discrete logarithm.

Although the Diffie-Hellman public key-distribution system is the first public-key cryptographic technique ever proposed, it is still generally considered to be one of the best.

9.11 Trapdoor One-Way Functions, Public-Key Cryptosystems and Digital Signatures

The second crucial definition used by Diffie and Hellman is considerably more subtle than that of a one-way function. A *trapdoor one-way function* as defined by Diffie and Hellman is actually a *family* of invertible (or "bijective") functions f_z, indexed by a parameter z (the "trapdoor"), such that (1) when z is known, it is easy to find *algorithms* E_z and D_z that easily compute f_z and f_z^{-1}, respectively, but

(2) for virtually all z and virtually any y in the range of f_z, it is computationally infeasible to find the x such that $f_z(x) = y$ when z is not known, even if the algorithm E_z is known. For such a function to be cryptographically useful, one needs to add to the Diffie-Hellman condition the additional condition that (3) it is easy to pick a value of z at random.

The cryptographic utility of a trapdoor one-way function is readily apparent; it can be used to create what Diffie and Hellman called a *Public-key cryptosystem* (PKC). Each user in a network secretly and randomly chooses a value of the trapdoor, say user i chooses z_i. This is easy to do because of condition (3) above. User i then computes his *decryption algorithm* D_{z_i}; he *also* computes the corresponding encryption algorithm E_{z_i} that he then places in the Trusted Public Directory (TPD) [after satisfying the custodian of their TPD that he is indeed user i]. Usually, the algorithms E_{z_i} and D_{z_i} are known in advance except for the values of certain parameters; it suffices then for user i to place the parameters of E_{z_i} in the TPD and user i need only to store the parameters of D_{z_i} to be able to use his decryption algorithm. The parameters of E_{z_i} are called the *public key* of user i, while the parameters of D_{z_i} are called the *private key* of user i.

Should user j wish to send a secret message x to user i, he merely fetches the public key of user i from the TPD, then uses the algorithm E_{z_i} to compute the cryptogram $y = f_{z_i}(x)$. User i can decipher this message since his private key permits him to use the algorithm D_{z_i} to compute $f_{z_i}^{-1}(y) = x$. But if f_z is truly a trapdoor one-way function, it will be computationally infeasible for any other user to decipher this message intended for user i.

Diffie and Hellman also showed in 1976 that, provided the domain and the range of the trapdoor one-way function f_Z coincide for every z (in which case f_Z is called a *trapdoor one-way permutation*), f_Z can also be used to ensure the authenticity of messages or, as they more colorfully described it, to create *digital signatures*. Here Diffie and Hellman assumed that *legitimate messages have sufficient internal structure to be reliably distinguished from random sequences*. If f_Z is a trapdoor one-way permutation and user j wishes to sign a message x so that it is unmistakable that it came from him, user j applies his decryption algorithm to x to compute the *signed message* $x' = f_{z_j}^{-1}(x)$. Any other user, say user i, who obtains x' can fetch user j's public key from the TPD, then use algorithm E_{z_j} to compute $f_{z_j}(x') = x$. However, only user j knows how to write the meaningful message x as the random-looking string x'; it is computationally infeasible for any other user when given x to find the x' such that $f_{z_j}(x') = x$. Of course, the scheme as just described provides no secrecy. If secrecy is also desired, user j [after forming $x' = f_{z_j}^{-1}(x)$] could fetch the public key of user i from the TPD, then use the algorithm E_{z_i} to compute the cryptogram $y' = f_{z_i}(x')$. User i would first decrypt this cryptogram y' with the help of his own private key, then he could check its authenticity with the help of user j's public key.

There are many cryptographers who think that the digital-signature capability of trapdoor one-way permutations is the most important contribution of public-key cryptography.

9.12 The Rivest-Shamir-Adleman (RSA) (Conjectured) Trapdoor One-Way Function

Although Diffie and Hellman shrewdly defined trapdoor one-way functions in their 1976 paper and clearly pointed out the cryptographic potential of such functions, they did not even hazard a conjecture as to a possible such function. The first such proposal was made in 1978 by the M.I.T. researchers R. L. Rivest,

A. Shamir and L. Adleman (RSA for short). The RSA "function" is the family of functions f_z where

$$z = (p_1, p_2, e), \tag{9.33}$$

in which p_1 and p_2 are *distinct odd primes* (whose product we denote by $m = p_1 p_2$) and in which e is a *unit* of $\mathbb{Z}_{\varphi(m)}$, and where $f_z : \mathbb{Z}_m \to \mathbb{Z}_m$ is the e-th *power function*

$$f_z(x) = x^e \tag{9.34}$$

in $\mathbb{Z}_m$, the ring of integers modulo $m = p_1 p_2$. Because the domain and range of f_z coincide for all z, the precise *RSA conjecture* is that f_z is a *trapdoor one-way permutation*, provided that p_1 and p_2 are both large primes (RSA suggested that they have about 100 decimal digits) each such that $p_1 - 1$ and $p_2 - 1$ both have large prime factors, and that e is chosen uniformly at random in $\mathbb{Z}^*_{\varphi(m)}$. [The choice $e = 1$ is obviously bad and the choice $e = -1$ is not much better, but there seems to be no other especially bad choice for e. Nonetheless, it is to err on the safe side to make the random choice of e that RSA recommend.]

We now have before us the following tasks: *First*, we must show that when z is known, it is easy to find an algorithm E_z that easily computes f_z. *Second*, we must show that f_z is indeed an invertible (or bijective) function. *Third*, we must show that, when z is known, it is easy to find an algorithm D_z that easily computes f_z^{-1}. *Fourth*, we must make it at least plausible that, for virtually all $z = (p_1, p_2, e)$ and virtually all $y \in \mathbb{Z}_m$, it is computationally infeasible to find the $x \in \mathbb{Z}_m$ such that $y = x^e$, even when E_z is known. *Fifth* and finally, we must show that it is easy to choose the trapdoor $z = (p_1, p_2, e)$ at random.

When the trapdoor $z = (p_1, p_2, e)$ is known, one can easily multiply p_1 and p_2 to get the modulus $m = p_1 p_2$. Thus, one easily obtains the parameters m and e needed to compute the power function (9.34) by the square-and-multiply algorithm, which is our desired algorithm E_z for easily computing f_z. The parameter pair (m, e) needed to specify E_z is the *public key* of the RSA system.

When $z = (p_1, p_2, e)$ is known, one can easily compute $\varphi(m) = \varphi(p_1 p_2) = (p_1 - 1)(p_2 - 1)$, as follows from (7.36). Because e is a unit of $\mathbb{Z}_{\varphi(m)}$, we can then easily find its multiplicative inverse $d = e^{-1}$ in $\mathbb{Z}_{\varphi(m)}$ by using, say, Euclid's (or Stein's) extended greatest common divisor algorithm with inputs $\varphi(m)$ and e, as was shown in Section 7.5. We now claim that the power function

$$g_z(y) = y^d \tag{9.35}$$

in $\mathbb{Z}_m$ is in fact the desired inverse function f_z^{-1}. If this claim is true, then we can take the square-and-multiply algorithm to be our easy-to-find algorithm for easily computing f_z^{-1} and we can take the parameter (m, d) to be the *private key* of the RSA system. Our task thus reduces to showing that

$$(x^e)^d = x \tag{9.36}$$

holds for all x in $\mathbb{Z}_m$. Note that $de = Q\varphi(m) + 1$ in $\mathbb{Z}$, where Q is some integer, because $de = 1$ in $\mathbb{Z}_{\varphi(m)}$. Suppose first that x is a unit of $\mathbb{Z}_m$ (which actually covers all cases of real interest since otherwise either x is 0 or $\gcd(m, x)$ is either p_1 or p_2). Then

$$x^{de} = x^{Q\varphi(m)+1} = \left(x^{\varphi(m)}\right)^Q x = x$$

where we have made use of Euler's theorem from Section 7.8. Thus (9.36) holds when x is a unit of $\mathbb{Z}_m$, and it holds trivially when $x = 0$. It remains only to consider the case when x is a non-zero non-unit

of $\mathbb{Z}_m$, i.e., when x is divisible by either p_1 or p_2, but not both. With no loss of essential generality, suppose that p_1, but not p_2, divides x. Then the Chinese remainder theorem (CRT) transform (x_1, x_2) of x [see Section 7.6] for the modulus $m_1 = p_1$ and $m_2 = p_2$ is $(0, x_2)$ where $x_2 \neq 0$. The CRT transform of x^{de} is then $(0, x_2^{de})$ where x_2^{de} is computed in $\mathbb{Z}_{p_2} = GF(p_2)$, as follows from the CRT characterization of multiplication in $\mathbb{Z}_m$. But

$$x_2^{de} = x_2^{Q\varphi(m)+1} = \left(x_2^{\varphi(m)}\right)^Q x_2 = \left(x_2^{(p_2-1)}\right)^{(p_1-1)Q} x_2 = x_2$$

in $GF(p_2)$, where we have made use of Fermat's theorem from Section 7.8. Thus, the CRT transform of x^{de} is $(0, x_2)$, the same as the CRT transform of x. It now follows from the CRT that $x^{de} = x$, which completes the proof that (9.36) holds for all x in $\mathbb{Z}_m$, and hence that

$$f_z^{-1}(y) = y^d \tag{9.37}$$

in $\mathbb{Z}_m$ is indeed the inverse of the RSA function f_z.

We have now reached the fourth task of showing that, for virtually all z and virtually all y in $\mathbb{Z}_m$, it is at least plausibly computationally infeasible to find the x such that $y = f_z(x)$, even if E_z is known or, equivalently, *when the public key* (m, e) *is known*. If we could factor m, then of course we would find p_1 and p_2 and hence learn the trapdoor $z = (p_1, p_2, e)$. The RSA conjecture really reduces to the two conjectures that, first, any way of inverting f_z given the public key (m, e) is computationally equivalent to factoring m and, second, that factoring m is computationally infeasible when p_1 and p_2 are randomly chosen large primes such that $p_1 - 1$ and $p_2 - 1$ both have large prime factors. There is some hard (but incomplete) evidence for the first of these two conjectures. If we could find $\varphi(m)$ from (m, e) then we could easily compute $d = e^{-1}$ in $\mathbb{Z}_{\varphi(m)}$ and hence easily invert f_z. But finding $\varphi(m)$ also tells us the value of

$$\begin{aligned} -b &= m - \varphi(m) + 1 \\ &= p_1 p_2 - (p_1 - 1)(p_2 - 1) + 1 \\ &= p_1 + p_2 \end{aligned}$$

and we already know the value of

$$c = m = p_1 p_2$$

so that we can now easily find p_1 and p_2 by computing the two roots of the real quadratic equation

$$X^2 + bX + c = 0$$

with the aid of a simple integer square-root algorithm. Thus, breaking RSA by finding $\varphi(m)$ is computationally equivalent to factoring m. In fact, all attacks thus far proposed against RSA have turned out to be computationally equivalent to factoring m – but no proof has been forthcoming that this must be the case. There is no hard evidence, however, that factoring of integers is inherently difficult, but there is much historical evidence in support of this second conjecture. Anyone who has tried to improve upon known factoring algorithms soon convinces himself that this is a very hard problem. As of this writing in 1991, the best general factoring algorithms can factor a 100 decimal digit number in about 10 days (i.e., about 0.027 years) on the fastest supercomputer. Moreover, all fast general factoring algorithms have running times (either proved or conjectured) that grow with the size of the integer m to be factored in such a way that, when m has between about 50 and 200 decimal digits, the factoring time increases by a factor of 10 when m is increased by about 15 decimal digits (i.e., about 50 binary digits). Thus, it would

take about $(0.027) \times 10^{(200-100)/15} > 10^5$ years today to factor an m having about 200 decimal digits, which is the size recommended by RSA. The RSA system is safe from an attack made by factoring, at least until revolutionary advances in factoring algorithms have been made!

We now come to our final task, showing that it is easy to choose randomly the trapdoor $z = (p_1, p_2, e)$ for RSA. How to find easily the two very large primes needed is such an interesting story in itself that we postpone this discussion to the next section. When p_1 and p_2 have been obtained, it is easy to choose e uniformly random in $\mathbb{Z}^*_{\varphi(m)}$. One merely chooses e uniformly random in $\mathbb{Z}_{\varphi(m)}$, then checks whether e is indeed a unit by computing $\gcd(\varphi(m), e)$ with, say, Euclid's (or Stein's) gcd algorithm and checking to see if the result is 1. If not, one makes another random choice, etc.. The probability of success on any random choice is $\varphi(\varphi(m))/\varphi(m)$ which, because $p_1 - 1$ and $p_2 - 1$ have large prime factors, is large so that only a few random choices will be needed on the average. For instance, if $p_1 - 1 = 2\,p_1'$ and $p_2 - 1 = 2\,p_2'$ where p_1' and p_2' are also odd primes, the probability of success on any random choice is

$$\frac{\varphi(\varphi(p_1 p_2))}{\varphi(p_1 p_2)} = \frac{\varphi((p_1 - 1)\,(p_2 - 1))}{(p_1 - 1)\,(p_2 - 1)} = \frac{\varphi(4\,p_1' p_2')}{(p_1 - 1)\,(p_2 - 1)} = \frac{2\,(p_1' - 1)\,(p_2' - 1)}{4 p_1' p_2'} \approx \frac{1}{2}$$

so that only two random choices will be needed on the average to find e.

9.13 Finding Large Primes

According to the well-known number theorist H.W. Lenstra, Jr., "all modern primality testing rests on generalizations of Fermat's theorem". Fermat's theorem implies that when m is a prime number, then $R_m(b^{m-1}) = 1$ for any integer b, $2 \leq b < m$ (cf. Section 7.8). One concludes that a prime number must pass the following test.

Fermat's Test: Let b and m be integers and suppose $2 \leq b < m$. If

$$R_m(b^{m-1}) \neq 1, \tag{9.38}$$

then m *is not a prime.*

Note that $b = m - 1$, which equals $\ominus 1$ in $\mathbb{Z}_m$, always passes the Fermat test, i.e., if m is odd $R_m((-1)^{m-1}) = 1$. Therefore, one should always choose the base b within the interval $2 \leq b < m - 1$ for primality testing. If $R_m(b^{m-1}) = 1$, then one says that m is a *pseudoprime to the base* b (but m could also be a genuine prime.)

Example 9.13.1 Let $m = 91$. By choosing, e.g., $b = 2$ and by computing $R_{91}(2^{90}) = 64$, Fermat's test implies that 91 is not a prime.

Let $m = 1105$ and choose $b = 3$. Then, $R_{1105}(3^{1104}) = 1$ so that 1105 is a pseudo-prime to the base 3 and hence one can draw no conclusion as to whether m is a prime or not. It can be shown that the integer $m = 1105$ is a *Carmichael number*, i.e., it satisfies $R_m(b^{m-1}) = 1$ for all integers b, $2 \leq b < m$, which are relatively prime to m. Hence, by using only the Fermat test there is little chance to conclude that $m = 1105$ is composite by randomly choosing the base b in the interval $2 \leq b < 1105$. In fact, $R_m(b^{m-1}) \neq 1$ if and only if $\gcd(m, b) \neq 1$ and therefore, only $m - 2 - \varphi(m) = 335$ of all $m - 2 = 1103$ possible choices for b will *not* pass the Fermat test.

To overcome the weakness of the Fermat test shown in Example 9.13.1, one can generalize this test by imposing additional conditions relying on square root properties in the finite field $GF(p)$. If p is an odd

prime, then $1 \in GF(p)$ has the two square roots 1 and $\ominus 1$, i.e., if $x^2 = 1$ then $x = 1$ or $x = \ominus 1 = p - 1$ in $GF(p)$. From this fact, one obtains the following test that an integer m must pass, if it is a prime.

Square-Root Test: Let m be an odd integer at least 3 and suppose that $x \in \mathbb{Z}_m$ satisfies $x^2 = 1$. If $x \neq 1$ and $x \neq m - 1$, then m *is not a prime.*

We can combine the Fermat test and the square-root test by choosing $x = R_m(b^{(m-1)/2})$, where $2 \leq b < m - 1$. If the odd integer m passes the Fermat test, then $x^2 = R_m(b^{m-1}) = 1$, and one can apply the square-root test.

Example 9.13.2 Let $m = 1105$ and $b = 3$. From Example 9.13.1 we know that m passes the Fermat test, but it does not pass the square-root test because $x = R_m(b^{(m-1)/2}) = R_{1105}(3^{552}) = 781$. Hence, m is not prime (which is obvious because 5 clearly divides m).

If $x = R_m(b^{(m-1)/2}) = 1$ and $\frac{m-1}{2}$ is even, one can apply the square-root test again to $y = R_m(b^{(m-1)/4})$. Iterating this procedure, one obtains one of the most efficient and widely used primality test, viz. Miller's test, which is given in flowchart form in Fig. 9.6.

Miller's Test: Let m be an odd integer at least 3 and let b be a base satisfying $2 \leq b < m - 1$ and suppose $m - 1 = 2^s Q$, where Q is odd. If all of the following inequalities

$$R_m(b^Q) \neq 1,\ R_m(b^Q) \neq m - 1,\ R_m(b^{2Q}) \neq m - 1, \ldots, R_m(b^{2^{s-1}Q}) \neq m - 1$$

hold, then m *is not prime.*

If m passes Miller's test for a given b, it is called a *strong pseudoprime to the base b.* The efficiency of Miller's test is guaranteed by the following theorem that we state without proof.

Rabin's Theorem: If m is *not a prime*, then m can pass Miller's test for at most $\lfloor m/4 \rfloor$ possible choices of the base b, $2 \leq b < m$.

Corollary to Rabin's Theorem: The probability that m passes Miller's test for n independent random choices of the base b, given that m is not a prime, is less than or equal to $\frac{1}{4^n}$.

Thus, the two RSA primes can be found to any *desired confidence* by applying Miller's test for n randomly chosen bases. Using the square-and-multiply algorithm to compute $R_m(b^Q)$ and a check to avoid that m is divisible by some small prime number, one obtains the following *efficient* algorithm to find RSA primes.

Efficient Finding of "Probable Primes" for RSA:

Precomputation:

Compute $r \approx 10^{100}$ as $r = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot \ldots$ (product of successive smallest primes).

Computation:

1. Randomly choose an m in $1 \leq m \leq 10^{100}$ (perhaps eliminating some "dumb" choices) and check whether $\gcd(r, m) = 1$. If no, repeat this step (which requires only 333 substractions by Stein's algorithm).

2. Randomly choose a base b ($2 \leq b \leq m - 2$) and check whether m passes Miller's test. If no, return to step 1.

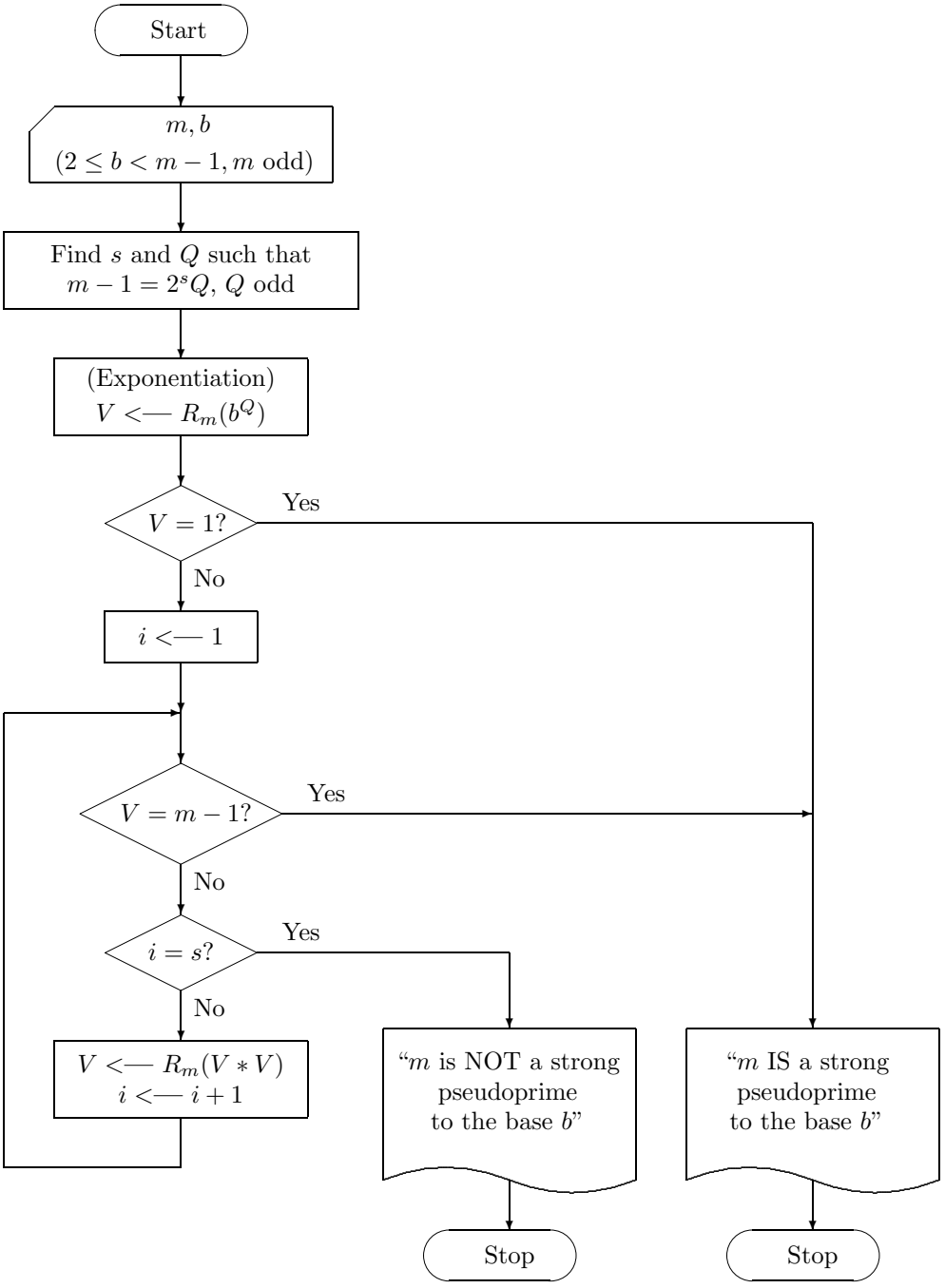

Figure 9.6: Flowchart of Miller's Test.

3. Repeat step 2 until you are convinced that m is a prime.

NOTE: One might wish simultaneously to *check both* m and $\frac{m-1}{2}$ for primeness to ensure that $p-1$ has a large prime factor.

Bibliography

The following textbooks and introductory papers are recommended supplementary reading.

Algebra

I. N. Herstein, *Topics in Algebra*, 2nd Ed., Wiley, 1975.

N. Jacobson, *Basic Algebra I*, Freeman & Co., San Francisco, 1974.

S. Lang, *Undergraduate Algebra*, 2nd Ed., Springer, 1990.

Algebraic Coding Theory

R. E. Blahut, *The Theory and Practice of Error Control Codes*, Addison-Wesley, 1984.

J. H. van Lint, *Introduction to Coding Theory*, GTM, Springer, 1982.

F. J. MacWilliams and N. J.. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, New York, 1977.

W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd Ed., MIT Press, 1972.

C. E. Shannon and W. Weaver, *The Mathematical Theory of Communications*, University of Illinois Press, 1963.

S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

Cryptography

C. E. Shannon, "Communication Theory of Secrecy Systems", *B.S.T.J*, Vol.28, pp. 656-715, Oct. 1949.

Contemporary Cryptology: The Science of Information Integrity (Ed. G. J. Simmons), IEEE Press, 1992.

A. J. Menezes, P. C. van Dorschof, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.