Math 422
Coding Theory

John C. Bowman
Lecture Notes

University of Alberta
Edmonton, Canada

January 27, 2003

Contents

List of Figures

Preface

These lecture notes are designed for a one-semester course on error-correcting codes and cryptography at the University of Alberta. I would like to thank my colleagues, Professors Hans Brungs, Gerald Cliff, and Ted Lewis, for their written notes and examples, on which these notes are partially based (in addition to the references listed in the bibliography).

Chapter 1

Introduction

In the modern era, digital information has become a valuable commodity. For example, the news media, governments, corporations, and universities all exchange enormous quantities of digitized information every day. However, the transmission lines that we use for sending and receiving data and the magnetic media (and even semiconductor memory devices) that we use to store data are imperfect.

Since transmission line and storage devices are not 100% reliable device, it has become necessary to develop ways of detecting when an error has occurred and, ideally, correcting it. The theory of error-correcting codes originated with Claude Shannon's famous 1948 paper "A Mathematical Theory of Communication" and has grown to connect to many areas of mathematics, including algebra and combinatorics. The cleverness of the error-correcting schemes that have been developed since 1948 is responsible for the great reliability that we now enjoy in our modern communications networks, computer systems, and even compact disk players.

Suppose you want to send the message "Yes" (denoted by 1) or "No" (denoted by 0) through a noisy communication channel. We assume that for there is a uniform probability $p < 1$ that any particular binary digit (often called a *bit*) could be altered, independent of whether or not any other bits are transmitted correctly. This kind of transmission line is called a *binary symmetric* channel. (In a *q-ary symmetric channel*, the digits can take on any of q different values and the errors in each digit occur independently and manifest themselves as the $q-1$ other possible values with equal probability.)

If a single bit is sent, a binary channel will be reliable only a fraction $1-p$ of the time. The simplest way of increasing the reliability of such transmissions is to send the message twice. This relies on the fact that, if p is small then the probability p^2 of two errors occurring, is very small. The probability of no errors occurring is $(1-p)^2$. The probability of one error occurring is $2p(1-p)$ since there are two possible ways this could happen. While reception of the original message is more likely than any other particular result if $p < 1/2$, we need $p < 1 - 1/\sqrt{2} \approx 0.29$ to be sure that the correct message is received most of the time.

If the message 11 or 00 is received, we would expect with conditional probability

$$1 - \frac{p^2}{(1-p)^2 + p^2} = \frac{(1-p)^2}{(1-p)^2 + p^2}$$

that the sent message was "Yes" or "No", respectively. If the message 01 or 10 is received we know for sure that an error has occurred, but we have no way of knowing, or even reliably guessing, what message was sent (it could with equal probability have been the message 00 or 11). Of course, we could simply ask the sender to retransmit the message; however this would now require a total of 4 bits of information to be sent. If errors are reasonably frequent, it would make more sense to send three, instead of two, copies of the original data in a single message. That is, we should send "111" for "Yes" or "000" for "No". Then, if only one bit-flip occurs, we can always guess, with good reliability what the original message was. For example, suppose "111" is sent. Then of the eight possible received results, the patterns "111", "011", "101", and "110" would be correctly decoded as "Yes". The probability of the first pattern occurring is $(1-p)^3$ and the probability for each of the next three possibilities is $p(1-p)^2$. Hence the probability that the message is correctly decoded is

$$(1-p)^3 + 3p(1-p)^2 = (1-p)^2(1+2p) = 1 - 3p^2 + 2p^3.$$

In other words, the probability of a decoding error, $3p^2 - 2p^3$, is small. This kind of data encoding is known as a *repetition code*. For example, suppose that $p = 0.001$, so that on average one bit in every thousand is garbled. Triple-repetition decoding ensures that only about one bit in every 330 000 is garbled.

1.A Error Detection and Correction

Despite the inherent simplicity of repetition coding, sending the entire message like this in triplicate is not an efficient means of error correction. Our goal is to find optimal encoding and decoding schemes for reliable error correction of data sent through noisy transmission channels.

The sequences "000" and "111" in the previous example are known as *binary codewords*. Together they comprise a *binary code*. More generally, we make the following definitions.

Definition: Let $q \in \mathbb{Z}$. A q-ary *codeword* is a finite sequence of symbols, where each symbol is chosen from the *alphabet* (set) $F_q = \{\lambda_1, \lambda_2, \dots, \lambda_q\}$. Typically, we will take F_q to be the set $\mathbb{Z}_q \doteq \{0, 1, 2, \dots, q-1\}$. (We use the symbol $\doteq$ to emphasize a definition, although the notation := is more common.) The codeword itself can be thought of as a vector in the space $F_q^n = \underbrace{F_q \times F_q \times \dots F_q}_{n \text{ times}}$.

- A binary codeword, corresponding to the case $q = 2$, is just a finite sequence of 0s and 1s.

Definition: A q-ary *code* is a set of M codewords, where $M \in \mathbb{N}$ is known as the *size* of the code.

- The set of all words in the English language is a code over the 26-letter alphabet $\{A, B, \ldots, Z\}$.

One important aspect of all error-correcting schemes is that the extra information that accomplishes this must itself be transmitted and is hence subject to the same kinds of errors as is the data. So there is no way to **guarantee** accuracy; one just attempts to make the probability of accurate decoding as high as possible. Hence, a good code is one in which the codewords have little resemblance to each other. If the codewords are sufficiently different, we will soon see that it is possible not only to detect errors but even to correct them, using *nearest-neighbour decoding*, where one maps the received vector back to the closest nearby codeword.

- The set of all 10-digit telephone numbers in the United Kingdom is a 10-ary code of length 10. It is possible to use a code of over 82 million 10-digit telephone numbers (enough to meet the needs of the U.K.) such that if just one digit of any phone number is misdialled, the correct connection can still be made. Unfortunately, little thought was given to this, and as a result, frequently misdialled numbers do occur in the U.K. (as well as in North America!).

Definition: We define the *Hamming distance* $d(x, y)$ between two codewords x and y of F_q^n as the number of places in which they differ.

Remark: Notice that $d(x, y)$ is a *metric* on F_q^n since it is always non-negative and satisfies

1. $d(x, y) = 0 \iff x = y$,
2. $d(x, y) = d(y, x)$ for all $x, y \in F_q^n$,
3. $d(x, y) \le d(x, z) + d(z, y)$ for all $x, y, z \in F_q^n$.

The first two properties are immediate consequences of the definition, while the third property is known as the *triangle inequality*. It follows from the simple observation that $d(x, y)$ is the minimum number of digit changes required to change x to y. However, if we change x to y by first changing x to z and then changing z to y, we require $d(x, z) + d(z, y)$ changes. Thus $d(x, y) \le d(x, z) + d(z, y)$.

Remark: We can use property 2 to rewrite the triangle inequality as

$$d(x, y) - d(y, z) \le d(x, z) \quad \forall x, y, z \in F_q^n.$$

Definition: The *weight* $w(x)$ of a binary codeword x is the number of nonzero digits it has.

Remark: Let x and y be binary codewords in $\mathbb{Z}_2^n$. Then $d(x,y) = w(x-y) = w(x) + w(y) - 2w(xy)$. Here, $x-y$ and xy are computed mod 2, digit by digit.

Remark: Let x and y be codewords in $\mathbb{Z}_q^n$. Then $d(x,y) = w(x-y)$. Here, $x-y$ is computed mod q, digit by digit.

Definition: Let C be a code in F_q^n. We define the *minimum distance* $d(C)$ of the code to be

$$d(C) = \min\{d(x,y) : x, y \in F_q^n, x \neq y\}.$$

Remark: In view of the previous discussion, a good code is one with a relatively large minimum distance.

Definition: An (n, M, d) code is a code of length n, containing M codewords and having minimum distance d.

- For example, here is a $(5, 4, 3)$ code, consisting of four codewords from F_2^5, which are at least a distance 3 from each other.

$$C_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Upon considering each of the $\binom{4}{2} = \frac{4\times 3}{2} = 6$ pairs of distinct codewords (rows), we see that the minimum distance of C_3 is indeed 3. With this code, we can either (i) detect up to two errors (since the members of each pair of distinct codewords are more than a distance 2 apart), or (ii) detect and correct a single error (since, if only a single error has occurred, the received vector will still be closer to the transmitted codeword than to any other).

The following theorem shows how this works in general.

Theorem 1.1 (Error Detection and Correction) *In a symmetric channel with error-probability $p > 0$,*

(i) a code C can detect up to t errors in every codeword $\iff d(C) \geq t+1$;

(ii) a code C can correct up to t errors in any codeword $\iff d(C) \geq 2t+1$.

Proof:

(i) "$\Rightarrow$" Suppose $d(C) \geq t+1$. Suppose a codeword x is transmitted and t or fewer errors are introduced, resulting in a new vector $y \in F_q^n$. Then $d(x,y) = w(x-y) \leq t < t+1 = d(C)$, so the received codeword cannot be another codeword. Hence errors can be detected.

"$\Leftarrow$" Likewise, if $d(C) < t+1$, then there is some pair of codewords x and y that have distance $d(x,y) \leq t$. Since it is possible to send the codeword x and receive the codeword y by the introduction of t errors, we conclude that C cannot detect t errors.

(ii) Suppose $d(C) \geq 2t+1$. Suppose a codeword x is transmitted and t or fewer errors are introduced, resulting in a new vector $y \in F_q^n$ satisfying $d(x,y) \leq t$. If x' is a codeword other than x then $d(x,x') \geq 2t+1$ and the triangle inequality $d(x,x') \leq d(x,y) + d(y,x')$ implies that

$$d(y,x') \geq d(x,x') - d(x,y) \geq 2t+1-t = t+1 > t \geq d(y,x).$$

Hence the received vector y is closer to x than to any other codeword x', making it possible to identify the original transmitted codeword x correctly.

Likewise, if $d(C) < 2t+1$, then there is some pair of codewords x and x' that have distance $d(x,x') \leq 2t$. If $d(x,x') \leq t$, let $y = x'$. Otherwise, if $t < d(x,x') \leq 2t$, construct a vector y from x by changing t of the digits of x that are in disagreement with x' to their corresponding values in x'. In this way we construct a vector y such that $0 < d(y,x') \leq t < d(y,x)$. It is possible to send the codeword x and receive the vector y because of the introduction of t errors, and this would **not** be correctly decoded as x by using nearest-neighbour decoding.

Corollary 1.1.1 *If a code C has minimum distance d, then C can be used either (i) to detect up to $d-1$ errors or (ii) to correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors in any codeword. Here $\lfloor x \rfloor$ represents the greatest integer less than or equal to x.*

A good (n,M,d) code has small n (for rapid message transmission), large M (to maximize the amount of information transmitted), and large d (to be able to correct many errors. A main problem in coding theory is to find codes that optimize M for fixed values of n and d.

Definition: Let $A_q(n,d)$ be the largest value of M such that there exists a q-ary (n,M,d) code.

- Since we have already constructed a $(5,4,3)$ code, we know that $A_2(5,3) \geq 4$. We will soon see that 4 is in fact the maximum possible value of M; i.e. $A_2(5,3) = 4$.

To help us tabulate $A_q(n,d)$, let us first consider the following special cases:

Theorem 1.2 (Special Cases) *For any values of q and n,*

(i) $A_q(n, 1) = q^n$*;*

(ii) $A_q(n, n) = q$*.*

Proof:

(i) When the minimum distance $d = 1$, we require only that the codewords be distinct. The largest code with this property is the whole of F_q^n, which has $M = q^n$ codewords.

(ii) When the minimum distance $d = n$, we require that any two distinct codewords differ in all n positions. In particular, this means that the symbols appearing in the first position must be distinct, so there can be no more than q codewords. A q-ary repetition code of length n is an example of an (n, q, n) code, so the bound $A_q(n, n) = q$ can actually be realized.

Remark: There must be more at least two codewords for $d(C)$ even to be defined. This means that $A_q(n, d)$ is not defined if $d > n$, since $d(x, y) = w(x - y) \leq n$ for distinct codewords $x, y \in F_q^n$.

Lemma 1.1 (Reduction Lemma) *If a q-ary (n, M, d) code exists, there also exists an $(n - 1, M, d - 1)$ code.*

Proof: Given an (n, M, d) code, let x and y be codewords such that $d(x, y) = d$ and choose any column where x and y differ. Delete this column from all codewords. The result is an $(n - 1, M, d - 1)$ code.

Theorem 1.3 (Even Values of d) *Suppose d is even. Then a binary (n, M, d) code exists $\iff$ a binary $(n - 1, M, d - 1)$ code exists.*

Proof:

"$\Rightarrow$" This follows from Lemma 1.1.

"$\Leftarrow$" Suppose C is a binary $(n - 1, M, d - 1)$ code. Let $\hat{C}$ be the code of length n obtained by extending each codeword x of C by adding a parity bit $w(x) \pmod 2$. This makes the weight $w(\hat{x})$ of every codeword $\hat{x}$ of $\hat{C}$ even. Then $d(x, y) = w(x) + w(y) - 2w(xy)$ must be even for every codewords x and y in $\hat{C}$, so $d(\hat{C})$ is even. Note that $d - 1 \leq d(\hat{C}) \leq d$. But $d - 1$ is odd, so in fact $d(\hat{C}) = d$. Thus $\hat{C}$ is a (n, M, d) code.

Corollary 1.3.1 (Maximum code size for even d) *If d is even, then* $A_2(n, d) = A_2(n - 1, d - 1)$*.*

n	$d=3$	$d=5$	$d=7$
5	4	2	
6	8	2	
7	16	2	2
8	20	4	2
9	40	6	2
10	72-79	12	2
11	144-158	24	4
12	256	32	4
13	512	64	8
14	1024	128	16
15	2048	256	32
16	2560–3276	256–340	36–37

Table 1.1: Maximum code size $A_2(n,d)$ for $n \leq 16$ and $d \leq 7$.

This result means that we only need to calculate $A_2(n,d)$ for odd d. In fact, in view of Theorem 1.1, there is little advantage in considering codes with even d if the goal is error correction. In Table 1.1, we present values of $A_2(n,d)$ for $n \leq 16$ and for odd values of $d \leq 7$.

As an example, we now compute the value $A_2(5,3)$ entered in Table 1.1, after establishing a useful simplification, beginning with the following definition.

Definition: Two q-ary codes are *equivalent* if one can be obtained from the other by a combination of

(A) permutation of the columns of the code;

(B) relabelling the symbols appearing in a fixed column.

Remark: Note that the distances between codewords are unchanged by each of these operations. That is, equivalent codes have the same (n, M, d) parameters and will correct the same number of errors. Furthermore, in a q-ary symmetric channel, the error-correction performance of equivalent codes will be identical.

- The binary code

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

is seen to be equivalent to our previous $(5,4,3)$ code C_3 by switching columns 1 and 2 and then applying the permutation $0 \leftrightarrow 1$ to the first and fourth columns of the resulting matrix.

Lemma 1.2 (Zero Vector) *Any code over an alphabet containing the symbol* 0 *is equivalent to a code containing the zero vector* $\mathbf{0}$.

Proof: Given a code of length n, choose any codeword $x_1 x_2 \ldots x_n$. For each i such that $x_i \neq 0$, apply the permutation $0 \leftrightarrow x_i$ to the symbols in the ith column.

- Armed with the above lemma and the concept of equivalence, it is now easy to prove that $A_2(5,3) = 4$. Let C be a $(5, M, 3)$ code with $M \geq 4$. Without loss of generality, we may assume that C contains the zero vector (if necessary, by replacing C with an equivalent code). Then there can be no codewords with just one or two 1s, since $d = 3$. Also, there can be at most one codeword with four or more 1s; otherwise there would be two codewords with at least three 1s in common positions and less than a distance 3 apart. Since $M \geq 4$, there must be at least two codewords containing exactly three 1s. By rearranging columns, if necessary, we see that the code contains the codewords

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

  There is no way to add any more codewords containing exactly three 1s and we can also now rule out the possibility of five 1s. This means that there can be at most four codewords, that is, $A_2(5,3) \leq 4$. Since we have previously shown that $A_2(5,3) \geq 4$, we deduce that $A_2(5,3) = 4$.

Remark: A fourth codeword, if present in the above code, must have exactly four 1s. The only possible position for the 0 symbol is in the middle position, so the fourth codeword must be 11011. We then see that the resulting code is equivalent to C_3 and hence $A_2(5,3)$ is unique, up to equivalence.

The above trial-and-error approach becomes impractical for large codes. In some of these cases, an important bound, known as the *sphere-packing* or *Hamming bound*, can be used to establish that a code is the largest possible for given values of n and d.

Lemma 1.3 (Counting) *A sphere of radius* t *in* F_q^n, *with* $0 \leq t \leq n$, *contains exactly*

$$\sum_{k=0}^{t} \binom{n}{k} (q-1)^k$$

vectors.

Proof: The number of vectors that are a distance k from a fixed vector in F_q^n is $\binom{n}{k}(q-1)^k$, because there are $\binom{n}{k}$ choices for the k positions that differ from those of the fixed vector and there are $q-1$ values that can be assigned independently to each of these k positions. Summing over the possible values of k, we obtain the desired result.

Theorem 1.4 (Sphere-Packing Bound) *A q-ary $(n, M, 2t+1)$ code satisfies*

$$M \sum_{k=0}^{t} \binom{n}{k} (q-1)^k \le q^n. \tag{1.1}$$

Proof: By the triangle inequality, any two spheres of radius t that are centered on distinct codewords will have no vectors in common. The total number of vectors in the M spheres of radius t centered on the M codewords is thus given by the left-hand side of the above inequality; this number can be no more than the total number q^n of vectors in F_q^n.

- For our $(5, 4, 3)$ code, Eq. (1.1) gives the bound $M(1 + 5) \le 2^5 = 32$ which implies that $A_2(5, 3) \le 5$. We have already seen that $A_2(5, 3) = 4$. This emphasizes, that just because some set of numbers n, M, and d satisfy Eq. (1.1), there is no guarantee that such a code actually exists.

Definition: A *perfect code* is a code for which equality occurs in 1.1. For such a code, the M spheres of radius t centered on the codewords fill the whole space F_q^n completely, without overlapping.

Remark: Codes which consist of a single codeword (taking $t = n$) and codes which contain all vectors of F_q^n, along with the q-ary repetition code of length n are *trivially perfect codes.*

1.B Balanced Block Designs

Definition: A *balanced block design* consists of a collection of b subsets, called *blocks*, of a set S containing v *points* such that, for some fixed r, k, and λ:

(i) each point lies in exactly r blocks;

(ii) each block contains exactly k points;

(iii) each pair of points occurs together in exactly λ blocks.

Such a design is called a (b, v, r, k, λ) design.

- Let $S = \{1, 2, 3, 4, 5, 6, 7\}$ and consider the subsets $\{1, 2, 4\}$, $\{2, 3, 5\}$, $\{3, 4, 6\}$, $\{4, 5, 7\}$, $\{5, 6, 1\}$, $\{6, 7, 2\}$, $\{7, 1, 3\}$ of S. Each number lies in exactly 3 blocks, each block contains 3 numbers, and each pair of numbers occur together in exactly 1 block. The six lines and circle in Fig. 1.1 illustrate these relationships. Hence these subsets form a $(7, 7, 3, 3, 1)$ design.

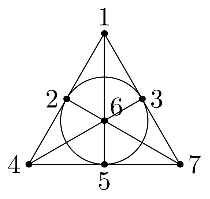

Figure 1.1: Seven-point plane.

Remark: The parameters (b, v, r, k, λ) are not independent. Consider the set of ordered pairs

$$T = \{(x, B) : x \text{ is a point}, B \text{ is a block}, x \in B\}.$$

Since each of the v points lie in r blocks, there must be a total of vr ordered pairs in T. Alternatively, we know that since there are b blocks and k points in each block, we can form exactly bk such pairs. Thus $bk = vr$. Similarly, by considering the set

$$U = \{(x, y, B) : x, y \text{ are distinct points}, B \text{ is a block}, x, y \in B\},$$

we deduce

$$b\frac{k(k-1)}{2} = \lambda\frac{v(v-1)}{2},$$

which, using $bk = vr$, simplifies to $r(k-1) = \lambda(v-1)$.

Definition: A block design is *symmetric* if $v = b$ (and hence $k = r$), that is, the number of points and blocks are identical. For brevity, this is called a (v, k, λ) design.

Definition: The *incidence matrix* of a block design is a $v \times b$ matrix with entries

$$a_{ij} = \begin{cases} 1 & \text{if } x_i \in B_j, \\ 0 & \text{if } x_i \notin B_j, \end{cases}$$

where x_i, $i = 1, \ldots, v$ are the design points and B_j, $j = 1, \ldots, b$ are the design blocks.

- For our above $(7, 3, 1)$ symmetric design, the incidence matrix A is

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

- We now construct a $(7, 16, 3)$ binary code C consisting of the zero vector $\mathbf{0}$, the unit vector $\mathbf{1}$, the 7 rows of A, and the 7 rows of the matrix B obtained from A by the interchange $0 \leftrightarrow 1$:

$$C = \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \\ \\ \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \\ \boldsymbol{a}_4 \\ \boldsymbol{a}_5 \\ \boldsymbol{a}_6 \\ \boldsymbol{a}_7 \\ \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \\ \boldsymbol{b}_5 \\ \boldsymbol{b}_6 \\ \boldsymbol{b}_7 \end{pmatrix} = \begin{pmatrix} 0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1 \\ \\ 1&0&0&0&1&0&1 \\ 1&1&0&0&0&1&0 \\ 0&1&1&0&0&0&1 \\ 1&0&1&1&0&0&0 \\ 0&1&0&1&1&0&0 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&0&1&1 \\ \\ 0&1&1&1&0&1&0 \\ 0&0&1&1&1&0&1 \\ 1&0&0&1&1&1&0 \\ 0&1&0&0&1&1&1 \\ 1&0&1&0&0&1&1 \\ 1&1&0&1&0&0&1 \\ 1&1&1&0&1&0&0 \end{pmatrix}.$$

To find the minimum distance of this code, note that each row of A has exactly three 1s and, by construction, any two distinct rows of A have exactly one 1 in common. Hence $d(\boldsymbol{a}_i, \boldsymbol{a}_j) = 3 + 3 - 2(1) = 4$ for $i \neq j$. Likewise, $d(\boldsymbol{b}_i, \boldsymbol{b}_j) = 4$. Furthermore,

$$d(0, a_i) = 3, \qquad d(0, b_i) = 4,$$

$$d(1, a_i) = 4, \qquad d(1, b_i) = 3,$$

$$d(a_i, b_i) = d(0, 1) = 7,$$

for $i = 1, \ldots, 7$. Finally, a_i and b_j disagree in precisely those places where a_i and a_j agree, so

$$d(a_i, b_j) = 7 - d(a_i, a_j) = 7 - 4 = 3, \qquad \text{for } i \neq j.$$

Thus C is a $(7, 16, 3)$ code, which in fact is perfect, since the equality in Eq. (1.1) is satisfied:

$$16\left(\binom{7}{0} + \binom{7}{1}\right) = 16(1 + 7) = 128 = 2^7.$$

The existence of a perfect binary $(7, 16, 3)$ code establishes $A_2(7, 3) = 16$, so we have now established another entry of Table 1.1.

1.C The ISBN code

Modern books are assigned an International Standard Book Number (ISBN), a 10-digit codeword, by the publisher. For example, Hill [1997] has the ISBN number 0-19-853803-0. Note that three hyphens separate the codeword into four fields. The first field specifies the language (0 means English), the second field indicates the publisher (19 means Oxford University Press), the third field (853803) is the the book number assigned by the publisher, and the final digit (0) is a check digit. If the digits of the ISBN number is denoted $\boldsymbol{x} = x_1 \dots x_{10}$, then the check digit x_9 is chosen as

$$x_{10} = \sum_{k=1}^{9} kx_k \pmod{11}.$$

If x_{10} turns out to be 10, an X is printed in place of the final digit. The tenth digit serves to make the *weighted check sum*

$$\sum_{k=1}^{10} kx_k = \sum_{k=1}^{9} kx_k + 10\sum_{k=1}^{9} kx_k = 11\sum_{k=1}^{9} kx_k = 0 \pmod{11}.$$

So, if $\sum_{k=1}^{10} kx_k \neq 0 \pmod{11}$, we know that an error has occurred. In fact, the ISBN number is able to (ii) detect a single error or (ii) detect a transposition error that results in two digits (not necessarily adjacent) being interchanged.

If a single error occurs, then some digit x_j is received as $x_j + e$ with $e \neq 0$. Then $\sum_{k=1}^{10} kx_k + je = je \pmod{11} \neq 0 (\bmod 11)$ since j and e are nonzero.

Let $\boldsymbol{y}$ be the vector obtained by exchanging the digits x_j and x_k in an ISBN code $\boldsymbol{x}$, where $j \neq k$. Then

$$\begin{aligned}\sum_{i=1}^{10} ix_i + (k-j)x_j + (j-k)x_k &= (k-j)x_j + (j-k)x_k \pmod{11}\\ &= (k-j)(x_j - x_k) \pmod{11} \neq 0 \pmod{11}\end{aligned}$$

if $x_j \neq x_k$.

In the above arguments we have used the property of the field $\mathbb{Z}_{11}$ (the integers modulo 11) that the product of two nonzero elements is always nonzero (that is, $ab = 0$ and $a \neq 0 \Rightarrow a^{-1}ab = 0 \Rightarrow b = 0$). Consequently, $\mathbb{Z}_{ab}$ with $a, b > 1$ cannot be a field because the product $ab = 0 \pmod{ab}$, even though $a \neq 0$ and $b \neq 0$. Note also that there can be no inverse a^{-1} in $\mathbb{Z}_{ab}$, for otherwise $b = a^{-1}ab = a^{-1}0 = 0 \pmod{ab}$.

In fact, $\mathbb{Z}_p$ is a field $\iff$ p is prime. For this reason, the ISBN code is calculated in $\mathbb{Z}_{11}$ and not in $\mathbb{Z}_{10}$, where $2 \cdot 5 = 0 \pmod{n}$.

The ISBN code cannot be used to correct error unless we know *a priori* which digit is in error. To do this, we first need to construct a table of inverses modulo 11 using the Euclidean division algorithm. For example, let y be the inverse of 2 modulo 11. Then $2y = 1 \pmod{11}$ implies $2y = 11q+1$ or $1 = -11q+2y$ for some integers y and q. On dividing 11 by 2 as we would to show that $\gcd(11, 2) = 1$, we find $11 = 5 \cdot 2 + 1$ so that $1 = 11 - 5 \cdot 2$, from which we see that $q = -1$ and $y = -5 \pmod{11} = 6 \pmod{11}$ are solutions. Similarly, $3^{-1} = 4 \pmod{11}$ since $11 = 3 \cdot 3 + 2$ and $3 = 1 \cdot 2 + 1$, so $1 = 3 - 1 \cdot 2 = 3 - 1 \cdot (11 - 3 \cdot 3) = -1 \cdot 11 + 4 \cdot 3$. The complete table of inverses modulo 11 are shown in Table 1.2.

x	1	2	3	4	5	6	7	8	9	10
x^{-1}	1	6	4	3	9	2	8	7	5	10

Table 1.2: Inverses modulo 11.

Suppose that we detect an error and we know in addition that it is the digit x_j that is in error (and hence unknown). Then we can use our table of inverses to solve for the value of x_j, assuming all of the other digits are correct. Since

$$jx + \sum_{\substack{k=1\\k\neq j}}^{10} kx_k = 0 \pmod{11},$$

we know that

$$x = -j^{-1} \sum_{\substack{k=1\\k\neq j}}^{10} kx_k \pmod{11}.$$

For example, if we did not know the fourth digit x_4 of the ISBN 0-19-x53803-0, we would calculate

$$\begin{aligned} x_4 &= -4^{-1}(1 \cdot 0 + 2 \cdot 1 + 3 \cdot 9 + 5 \cdot 5 + 6 \cdot 3 + 7 \cdot 8 + 8 \cdot 0 + 9 \cdot 3 + 10 \cdot 0) \pmod{11} \\ &= -3(0 + 2 + 5 + 3 + 7 + 1 + 0 + 5 + 0) \pmod{11} = -3(1) \pmod{11} = 8, \end{aligned}$$

which is indeed correct.

Chapter 2

Linear Codes

An important class of codes are linear codes in the vector space F_q^n.

Definition: A *linear code* C is a code for which, whenever $u \in C$ and $v \in C$, then $\alpha u + \beta v \in C$ for all $\alpha, \beta \in F_q$. That is, C is a linear subspace of F_q^n.

Remark: The zero vector $\mathbf{0}$ automatically belongs to all linear codes.

Remark: A binary code C is linear $\iff$ it contains $\mathbf{0}$ and the sum of any two codewords in C is also in C.

Exercise: Show that the $(7, 16, 3)$ code developed in the previous chapter is linear.

Remark: A linear code C will always be a k-dimensional linear subspace of F_q^n for some integer k between 1 and n. A k-dimensional code C is simply the set of all linear combinations of k linearly independent codewords, called *basis vectors*. We say that these k basis codewords *generate* or *span* the entire code space C.

Definition: We say that a k-dimensional code in F_q^n is a $[n, k]$ code, or if we also wish to specify the minimum distance d, a $[n, k, d]$ code.

Remark: Note that a q-ary $[n, k, d]$ code is a (n, q^k, d) code. To see this, let the k basis vectors of a $[n, k, d]$ code be $\boldsymbol{u}_j$, for $j = 1, \ldots, k$. The q^k codewords are obtained as the linear combinations $\sum_{j=1}^k a_j \boldsymbol{u}_j$; there are q possible values for each of the k coefficients a_j. Note that

$$\sum_{j=1}^{k} a_j \boldsymbol{u}_j = \sum_{j=1}^{k} b_j \boldsymbol{u}_j \Rightarrow \sum_{j=1}^{k} (a_j - b_j) \boldsymbol{u}_j = 0 \Rightarrow a_j = b_j, \quad j = 1, \ldots k,$$

by the linear independence of the basis vectors, so the q^k generated codewords are distinct.

Remark: Not every (n, q^k, d) code is a q-ary $[n, k, d]$ code (it might not be linear).

Definition: Define the *minimum weight* of a code to be $w(C) = \min\{w(x) : x \in C\}$.

One of the advantage of linear codes is illustrated by the following lemma.

Lemma 2.1 (Distance of a Linear Code) *If C is a linear code in F_q^n, then $d(C) = w(C)$.*

Proof: There exist codewords x, y, and z such that $d(x, y) = d(C)$ and $w(z) = w(C)$. Then

$$d(C) \le d(z, 0) = w(z - 0) = w(z) = w(C) \le w(x - y) = d(x, y) = d(C),$$

so $w(C) = d(C)$.

Remark: Lemma 2.1 implies, for a linear code, that we only have to examine the weights of the $M - 1$ nonzero codewords in order to find the minimum distance. In contrast, for a general nonlinear code, we need to make $\binom{M}{2} = M(M - 1)/2$ comparisons (between all possible pairs of distinct codewords) to determine the minimum distance.

Definition: A $k \times n$ matrix with rows that are basis vectors for a linear $[n, k]$ code C is called a *generator matrix* of C.

- A q-ary repetition code of length n is an $[n, 1, n]$ code with generator matrix $[1\ 1\ \ldots\ 1]$.

Exercise: Show that the $(7, 16, 3)$ perfect code in Chapter 1 is a $[7, 4, 3]$ linear code (note that $2^4 = 16$) with generator matrix

$$\begin{bmatrix} \mathbf{1} \\ \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Remark: Linear q-ary codes are not defined unless q is a power of a prime (this is simply the requirement for the existence of the field F_q). However, lower-dimensional codes can always be obtained from linear q-ary codes by projection onto a lower-dimensional subspace of F_q^n. For example, the ISBN code is a subset of the 9-dimensional subspace of F_{11}^{10} consisting of all vectors perpendicular to the vector $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$; this is the space

$$\left\{(x_1 x_2 \ldots x_{10}) : \sum_{k=1}^{10} k x_k = 0 \pmod{11}\right\}.$$

However, not all vectors in this set (for example X-00-000000-1) are in the ISBN code. That is, the ISBN code is **not** a linear code.

For linear codes we must slightly restrict our definition of equivalence so that the codes remain linear (e.g., in order that the zero vector remains in the code).

Definition: Two linear q-ary codes are *equivalent* if one can be obtained from the other by a combination of

(A) permutation of the columns of the code;

(B) multiplication of the symbols appearing in a fixed column by a nonzero scalar.

Definition: A $k \times n$ matrix of rank k is in *reduced echelon form* (or *standard form*) if it can be written as

$$[\,1_k \,|\, A\,],$$

where 1_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$ matrix.

Remark: A generator matrix for a vector space can always be reduced to an equivalent reduced echelon form spanning the same vector space, by permutation of its rows, multiplication of a row by a non-zero scalar, or addition of one row to another. Note that any combinations of these operators with (A) and (B) above will generate equivalent linear codes.

Exercise: Show that the generator matrix for the $(7,16,3)$ perfect code in Chapter 1 can be written in reduced echelon form as

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

2.A Encoding and Decoding

A $[n,k]$ linear code C contains q^k codewords, corresponding to q^k distinct messages. We identify each message with a k-tuple

$$\boldsymbol{u} = [\,u_1 \quad u_2 \quad \ldots \quad u_k\,],$$

where the components u_i are elements of F_q. We can *encode* $\boldsymbol{u}$ by multiplying it on the right with the generator matrix G. This maps $\boldsymbol{u}$ to the linear combination $\boldsymbol{u}G$ of the codewords. In particular the message with components $u_i = \delta_{ik}$ gets mapped to the codeword appearing in the kth row of G.

- Given the message $[0,1,0,1]$ and the above generator matrix for our $(7,16,3)$ code, the encoded codeword

$$[0\ \ 1\ \ 0\ \ 1]\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{bmatrix}=[0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0]$$

  is just the sum of the second and fourth rows of G.

Definition: Let C be a linear code over F_q^n. Let $\boldsymbol{a}$ be any vector in F_q^n. The set $\boldsymbol{a}+C=\{\boldsymbol{a}+\boldsymbol{x}:\boldsymbol{x}\in C\}$ is called a *coset* of C.

Lemma 2.2 (Equivalent Cosets) *Suppose that $a+C$ is a coset of a linear code C and $b\in a+C$. Then*

$$b+C=a+C.$$

Proof: Since $b\in a+C$, then $b=a+x$ for some $x\in C$. Consider any vector $b+y\in b+C$, with $y\in C$. Then

$$b+y=(a+x)+y=a+(x+y)\in a+C,$$

so $b+C\subset a+C$. Furthermore $a=b+(-x)\in b+C$, so the same argument implies $a+C\subset b+C$. Hence $b+C=a+C$.

The following theorem from group theory states that F_q^n is just the union of q^{n-k} distinct cosets of a linear $[n,k]$ code C, each containing q^k elements.

Theorem 2.1 (Lagrange's Theorem) *Suppose C is an $[n,k]$ code in F_q^n. Then*

(i) every vector of F_q^n is in some coset of C;

(ii) every coset contains exactly q^k vectors;

(iii) any two cosets are either equivalent or disjoint.

Proof:

(i) $a=a+0\in a+C$ for every $a\in F_q^n$.

(ii) Since the mapping $\phi(x)=a+x$ is one-to-one, $|a+C|=|C|=q^k$. Here $|C|$ denotes the number of elements in C.

(iii) Let $a,b\in C$. Suppose that the cosets $a+C$ and $b+C$ have a common vector $v=a+x=b+y$, with $x,y\in C$. Then $b=a+(x-y)\in a+C$, so by Lemma 2.2 $b+C=a+C$.

Definition: The *standard array* (or *Slepian*) of a linear $[n, k]$ code C in F_q^n is a $q^{n-k} \times q^k$ array listing all the cosets of C. The first row consists of the codewords in C themselves, listed with $\mathbf{0}$ appearing in the first column. Subsequent rows are listed one a a time, beginning with a vector of minimal weight that has not already been listed in previous rows, such that the entry in the (i, j)th position is the sum of the entries in position $(i, 1)$ and position $(1, j)$. The vectors in the first column of the array are referred to as *coset leaders*.

- Let us revisit our linear $(5, 4, 3)$ code

$$C_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

with generator matrix

$$G_3 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

The standard array for C_3 is a 8×4 array of cosets listed here in three groups of increasing coset leader weight:

0 0 0 0 0	0 1 1 0 1	1 0 1 1 0	1 1 0 1 1
0 0 0 0 1	0 1 1 0 0	1 0 1 1 1	1 1 0 1 0
0 0 0 1 0	0 1 1 1 1	1 0 1 0 0	1 1 0 0 1
0 0 1 0 0	0 1 0 0 1	1 0 0 1 0	1 1 1 1 1
0 1 0 0 0	0 0 1 0 1	1 1 1 1 0	1 0 0 1 1
1 0 0 0 0	1 1 1 0 1	0 0 1 1 0	0 1 0 1 1
0 0 0 1 1	0 1 1 1 0	1 0 1 0 1	1 1 0 0 0
0 1 0 1 0	0 0 1 1 1	1 1 1 0 0	1 0 0 0 1

Remark: The last two rows of the standard array for C_3 could equally well have been written as

1 1 0 0 0	1 0 1 0 1	0 1 1 1 0	0 0 0 1 1
1 0 0 0 1	1 1 1 0 0	0 0 1 1 1	0 1 0 1 0

Definition: If the codeword $\boldsymbol{x}$ is sent, but the received vector is $\boldsymbol{y}$, we define the *error vector* $\boldsymbol{e} \doteq \boldsymbol{y} - \boldsymbol{x}$.

Remark: If no more than t errors have occurred, the coset leaders of weight t or less are precisely the error vectors that can be corrected. Recall that the code C_3, having minimum distance 3, can only correct one error. For the code C_3, as long as no more than one error has occurred, the error vector will have weight at most one. We can then decode the received vector by checking to see under which codeword it appears in the standard array, remembering that the codewords themselves are listed in the first row. For example, if $\boldsymbol{y} = 10111$ is received, we know that the error vector $\boldsymbol{e} = 00001$, and the transmitted codeword must have been $\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{e} = 10111 - 00001 = 10110$.

Remark: If two errors have occurred, one cannot determine the original vector with certainty, because in each row with coset leader weight 2, there are actually two vectors of weight 2. For a code with minimum distance $2t + 1$, the rows in the standard array of coset leader weight greater than t can be written in more than one way, as we have seen above. Thus, if 01110 is received, then either $01110 - 00011 = 01101$ or $01110 - 11000 = 10110$ could have been transmitted.

Remark: Let C be a binary $[n, k]$ linear code and α_i denote the number of coset leaders for C having weight i, where $i = 0, \dots, n$. If p is the error probability for a single bit, then the probability $P_{\text{corr}}(C)$ that a received vector is correctly decoded is

$$P_{\text{corr}}(C) = \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i}.$$

Remark: If C can correct t errors then the coset leaders of weight no more than t are unique and hence the total number of such leaders of weight i is $\alpha_i = \binom{n}{i}$ for $0 \le i \le t$. In particular, if $n = t$, then

$$P_{\text{corr}}(C) = \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} = (p + 1 - p)^n = 1;$$

such a code is able to correct all possible errors.

Remark: For $i > t$, the coefficients α_i can be difficult to calculate. For a perfect code, however, we know that every vector is within a distance t of some codeword. Thus, the error vectors that can be corrected by a perfect code are precisely those vectors of weight no more than t; consequently,

$$\alpha_i = \begin{cases} \binom{n}{i} & \text{for } 0 \le i \le t, \\ 0 & \text{for } i > t. \end{cases}$$

- For the code C_3, we see that $\alpha_0 = 1$, $\alpha_1 = 5$, $\alpha_2 = 2$, and $\alpha_3 = \alpha_4 = \alpha_5 = 0$. Hence

$$P_{\text{corr}}(C_3) = (1-p)^5 + 5p(1-p)^4 + 2p^2(1-p)^3 = (1-p)^3(1 + 3p - 2p^2).$$

For example, if $p = 0.01$, then $P_{\text{corr}} = 0.99921$ and $P_{\text{err}} \doteq 1 - P_{\text{corr}} = 0.00079$, more than a factor 12 lower than the raw bit error probability p. Of course, this improvement in reliability comes at a price: we must now send $n = 5$ bits for every $k = 2$ information bits. The ratio k/n is referred to as the *rate* of the code. It is interesting to compare the performance of C_3 with a code that sends two bits of information by using two back-to-back repetition codes each of length 5 and for which $\alpha_0 = 1$, $\alpha_1 = 5$, and $\alpha_2 = 10$. We find that P_{corr} for such a code is

$$[((1-p)^5 + 5p(1-p)^4 + 10p^2(1-p)^3]^2 = [(1-p)^3(1+3p+6p^2)]^2 = 0.99998$$

so that $P_{\text{err}} = 0.00002$. While this error rate is almost four times lower than that for C_3, bear in mind that the repetition scheme requires the transmission of twice as much data for the same number of information digits (i.e. it has half the rate of C_3).

2.B Syndrome Decoding

The standard array for our $(5, 4, 3)$ code had 32 entries; for a general code of length n, we will have to search through 2^n entries every time we wish to decode a received vector. For codes of any reasonable length, this is not practical. Fortunately, there is a more efficient alternative, which we now describe.

Definition: Let C be a $[n, k]$ linear code. The dual code $C^\perp$ of C in F_q^n is the set of all vectors that are orthogonal to every codeword of C:

$$C^\perp = \{\boldsymbol{v} \in F_q^n : \boldsymbol{v}\cdot\boldsymbol{u} = 0,\ \forall \boldsymbol{u} \in C\}.$$

Remark: The dual code $C^\perp$ is just the *null space* of G. That is,

$$\boldsymbol{v} \in C^\perp \iff G\boldsymbol{v}^t = \mathbf{0}$$

(where the superscript t denotes transposition). This just says that $\boldsymbol{v}$ is orthogonal to each of the rows of G. From linear algebra, we know that the space spanned by the k independent rows of G is a k dimensional subspace and the null space of G, which is just $C^\perp$, is an $n-k$ dimensional subspace.

Definition: Let C be a $[n, k]$ linear code. The $(n-k) \times n$ generator matrix H for $C^\perp$ is called a *parity-check matrix.*

Remark: The number $r = n - k$ corresponds to the number of parity check digits in the code and is known as the *redundancy* of the code.

Remark: A code C is completely specified by its parity-check matrix:

$$C = \{\boldsymbol{u} \in F_q^n : H\boldsymbol{u}^t = \boldsymbol{0}\}$$

since this is just the space of all vectors that are orthogonal to every vector in $C^{\perp}$. That is, $H\boldsymbol{u}^t = \boldsymbol{0} \iff \boldsymbol{u} \in C$.

Theorem 2.2 (Minimum Distance) *A linear code has minimum distance $d \iff d$ is the maximum number such that any $d-1$ columns of its parity-check matrix are linearly independent.*

Proof: Let C be a linear code and $\boldsymbol{u}$ be a vector such that $w(\boldsymbol{u}) = d(C) = d$. But

$$\boldsymbol{u} \in C \iff H\boldsymbol{u}^t = 0.$$

Since $\boldsymbol{u}$ has d nonzero components, we see that some d columns of H are linearly dependent. However, any $d-1$ columns of H must be linearly independent, or else there would exist a nonzero codeword in C with weight $d-1$.

- For a code with weight 3, Theorem 2.2 tells us that any two columns of its parity-check matrix must be linearly independent, but that some 3 columns are linearly dependent.

Definition: Given a linear code with parity-check matrix H, the column vector $H\boldsymbol{u}^t$ is called the *syndrome* of $\boldsymbol{u}$.

Lemma 2.3 *Two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are in the same coset $\iff$ they have the same syndrome.*

Proof:

$$(\boldsymbol{u} - \boldsymbol{v}) \in C \iff H(\boldsymbol{u} - \boldsymbol{v})^t = \boldsymbol{0} \iff H\boldsymbol{u}^t = H\boldsymbol{v}^t.$$

Remark: We thus see that is there is a one-to-one correspondence between cosets and syndromes. This leads to an alternative decoding scheme known as *syndrome decoding*. When a vector $\boldsymbol{u}$ is received, one computes the syndrome $H\boldsymbol{u}^t$ and compares it to the syndromes of the coset leaders. If the coset leader having the same syndrome is of minimal weight within its coset, we know the error vector for decoding $\boldsymbol{u}$.

To compute the syndrome for a code, we need only first determine the parity check matrix. The following lemma describes an easy way to construct the *standard form* of the parity-check matrix from the standard form generator matrix.

Lemma 2.4 *The $(n-k)\times n$ parity-check matrix H for an $[n,k]$ code generated by the matrix $G=[1_k\,|\,A]$, where A is a $k\times(n-k)$ matrix, is given by*

$$[\,-A^t\,|\,1_{n-k}\,]\,.$$

Proof: This follows from the fact that the rows of G are orthogonal to every row of H, in other words, that

$$GH^t=[\,1_k\quad A\,]\begin{bmatrix}-A\\1_{n-k}\end{bmatrix}=1_k(-A)+(A)1_{n-k}=-A+A=0,$$

the $k\times(n-k)$ zero matrix.

- A parity-check matrix H_3 for our $(5,4,3)$ code is

$$H_3=\begin{bmatrix}1&1&1&0&0\\1&0&0&1&0\\0&1&0&0&1\end{bmatrix}.$$

Remark: The syndrome $H\boldsymbol{e}^t$ of a binary error vector $\boldsymbol{e}$ is just the sum of those columns of H for which the corresponding entry in $\boldsymbol{e}$ is nonzero.

The following theorem makes it particularly easy to correct errors of unit weight. It will play a particularly important role for the Hamming codes discussed in the next chapter.

Theorem 2.3 *The syndrome of a vector which has a single error of m in the ith position is m times the ith column of H.*

Proof: Let $\boldsymbol{e}_i$ be the vector with the value m in the ith position and zero in all other positions. If the codeword $\boldsymbol{x}$ is sent and the vector $\boldsymbol{y}=\boldsymbol{x}+\boldsymbol{e}_i$ is received the syndrome $H\boldsymbol{y}^t=H\boldsymbol{x}^t+H\boldsymbol{e}_i^t=\mathbf{0}+H\boldsymbol{e}_i^t=H\boldsymbol{e}_i^t$ is just m times the ith column of H.

- For our $(5,4,3)$ code, if $\boldsymbol{y}=10111$ is received, we compute $H\boldsymbol{y}^t=001$, which matches the fifth column of H. Thus, the fifth digit is in error (assuming that only a single error has occurred), and we decode $\boldsymbol{y}$ to the codeword 10110, just as we deduced earlier using the standard array.

Remark: If the syndrome does not match any of the columns of H, we know that more than one error has occurred. We can still determine which coset the syndrome belongs to by comparing the computed syndrome with a table of syndromes of all coset leaders. If the corresponding coset leader has minimal weight within its coset, we are able to correct the error. To decode errors of weight greater than one we will need to construct a syndrome table, but this table, having only q^{n-k} entries, is smaller than the standard array, which has q^n entries.

Chapter 3

Hamming Codes

One way to construct perfect binary $[n, k]$ codes that can correct single errors is to ensure that every nonzero vector in F_2^{n-k} appears as a unique column of H. In this manner, the syndrome of every possible vector in F_2^n can be identified with a column of H, so that every vector in F_2^n is at most a distance one away from a codeword. This is called a *binary Hamming code*, which we now discuss in the general space F_q^n.

Remark: One can form $q-1$ distinct scalar multiples of any nonzero vector in F_q^r.

Definition: Given an integer $r \geq 2$, let $n = (q^r - 1)/(q - 1)$. The *Hamming code* $\mathrm{Ham}(r, q)$ is a linear code in F_q^n for which the columns of the $r \times n$ parity-check matrix H are the n distinct non-zero vectors of F_q^r with first nonzero entry equal to 1.

Remark: Not only are the columns of H distinct, all nonzero multiples of any two columns are also distinct. That is, any two columns of H are linearly independent. The total number of nonzero column multiples that can thus be formed is $n(q-1) = q^r - 1$. Including the zero vector, we see that H yields a total of q^r distinct syndromes, corresponding to all possible error vectors of unit weight in F_q^r.

- The columns of the parity-check matrix for the binary Hamming code $\mathrm{Ham}(r, 2)$ consists of all possible nonzero binary codewords of length r.

Remark: The columns of the parity-check matrix may be written in any order.

Remark: The dimension k of $\mathrm{Ham}(r, q)$ is given by

$$n - r = \frac{q^r - 1}{q - 1} - r.$$

Exercise: Show that the standard form of the parity-check matrix for a binary Hamming code can be obtained by simply rearranging its columns.

- A parity-check matrix for the one-dimensional code $\mathrm{Ham}(2, 2)$ is

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

  which can be written in standard form as

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

  The generator matrix is then seen to be $[1 \quad 1 \quad 1]$. That is, $\mathrm{Ham}(2, 2)$ is just the binary triple-repetition code.

- A parity-check matrix for the one-dimensional code $\mathrm{Ham}(3, 2)$ in standard form, is

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

Exercise: Show that this code is equivalent to the $(7, 16, 3)$ perfect code in Chapter 1.

Remark: An equivalent way to construct the binary Hamming code $\mathrm{Ham}(r, 2)$ is to consider all $n = 2^r - 1$ nonempty subsets of a set S containing r elements. Each of these subsets corresponds to a position of a code in F_2^n. A codeword can then be thought of as just a collection of nonzero subsets of S. Any particular element a of the set will appear in exactly half (i.e. in 2^{r-1} subsets) of all 2^r subsets of S, so that an even number of the $2^r - 1$ nonempty subsets, will contain a. This gives us a parity-check equation, which says that the sum of all digits corresponding to a subset containing a must be $0 \pmod 2$. There will be a parity-check equation for each of the r elements of S corresponding to a row of the parity-check matrix H. That is, each column of H corresponds to one of the subsets, with a 1 appearing in the ith position if the subset contains the ith element and 0 if it doesn't.

- The parity check matrix for $\mathrm{Ham}(3, 2)$ can be constructed by considering all possible nonempty subsets of $\{a, b, c\}$, each of which corresponds to one of the digits of a codeword $\boldsymbol{x} = x_1 x_2 \ldots x_7$ in F_2^7:

$$\begin{array}{cccc|ccc} & a & a & a & a & & \\ b & & b & b & & b & \\ c & c & & c & & & c \\ \hline x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array}$$

  Given any four binary *information digits* x_1, x_2, x_3, and x_4, there will be a unique codeword satisfying $H\boldsymbol{x} = 0$; the *parity-check digits* x_5, x_6, and x_7 can be determined from the three checksum equations corresponding to each of the elements a, b, and c:

$$a: \quad x_2 + x_3 + x_4 + x_5 = 0 \pmod 2,$$

$$b: \quad x_1 + x_3 + x_4 + x_6 = 0 \pmod 2,$$

and

$$c: \quad x_1 + x_2 + x_4 + x_7 = 0 \pmod 2.$$

For example, the vector $\boldsymbol{x} = 1100110$ corresponds to the collection

$$\{\{b, c\}, \{a, c\}, \{a\}, \{b\}\}.$$

Since there are an even number of as, bs, and cs in this collection, we know that $\boldsymbol{x}$ is a codeword.

Exercise: Show that two distinct codewords $\boldsymbol{x}$ and $\boldsymbol{y}$ that satisfy the above three parity check equations must differ in at least 3 places.

Remark: For binary Hamming codes, there is a distinct advantage in rearranging the parity-check matrix so that the columns, treated as binary numbers, are arranged in ascending order. The syndrome, interpreted in exactly the same way as a binary number, immediately tells us in which position a single error has occurred.

- We can write the parity-check matrix for Ham$(3, 2)$ in the *binary ascending form*

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

If the vector 1110110 is received, the syndrome is $[0, 1, 1]^t$, which corresponds to the binary number 3, so we know immediately that the a single error must have occurred in the third position, without even looking at H. Thus, the transmitted codeword was 1100110.

Remark: For nonbinary Hamming codes, we need to compare the computed syndrome with all nonzero multiples of the columns of the parity-check matrix.

- A parity-check matrix for Ham$(2, 3)$ is

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}.$$

If the vector 2020, which has syndrome $[2, 1]^t = 2[1, 2]^t$, is received and at most a single digit is in error, we see that an error of 2 has occurred in the last position and decode the vector as $x = y - e = 2020 - 0002 = 2021$.

- A parity-check matrix for Ham$(3,3)$ is

$$H=\begin{bmatrix} 0&0&0&0&1&1&1&1&1&1&1&1&1\\ 0&1&1&1&0&0&0&1&1&1&2&2&2\\ 1&0&1&2&0&1&2&0&1&2&0&1&2 \end{bmatrix}.$$

  If the vector 2000 0000 00001 is sent and at most a single error has occurred, then from the syndrome $[1,2,1]^t$ we see that an error of 1 has occurred in the second-last position, so the transmitted vector was 2000 0000 00021.

The following theorem establishes that Hamming codes can always correct single errors, as we saw in the above examples, and also that they are perfect.

Theorem 3.1 (Hamming Codes are Perfect) *Every* Ham(r,q) *code is perfect and has distance* 3.

Proof: Since any two columns of H are linearly independent, we know from Theorem 2.2 that Ham(r,q) has distance at least 3, so it can correct single errors. The distance cannot be any greater than 3 because the nonzero columns

$$\begin{bmatrix}0\\ \vdots\\ 0\\ 0\\ 1\end{bmatrix},\ \begin{bmatrix}0\\ \vdots\\ 0\\ 1\\ 0\end{bmatrix},\ \begin{bmatrix}0\\ \vdots\\ 0\\ 1\\ 1\end{bmatrix}$$

are linearly dependent.

Furthermore, we know that Ham(r,q) has $M=q^k=q^{n-r}$ codewords, so the sphere-packing bound

$$q^{n-r}(1+n(q-1))=q^{n-r}(1+q^r-1)=q^n$$

is perfectly achieved.

Corollary 3.1.1 (Hamming Size) *For any integer $r\geq 2$, we have $A_2(2^r-1,3)=2^{2^r-1-r}$.*

- Thus $A_2(3,3)=2$, $A_2(7,3)=16$, $A_2(15,3)=2^{11}=2048$, and $A_2(31,3)=2^{26}$.

Chapter 4

Golay Codes

We saw in the last chapter that the linear Hamming codes are nontrivial perfect codes.

Q. Are there any other nontrivial perfect codes?

A. Yes, two other linear perfect codes were found by Golay in 1949. In addition, several nonlinear perfect codes are known that have the same n, M, and d parameters as Hamming codes.

A necessary condition for a code to be perfect is that its n, M, and d values satisfy the sphere-packing bound

$$M \sum_{k=0}^{t} \binom{n}{k} (q-1)^k = q^n, \tag{4.1}$$

with $d = 2t + 1$. Golay found three other possible integer triples (n, M, d) that do not correspond to the parameters of a Hamming or trivial perfect codes. They are $(23, 2^{12}, 7)$ and $(90, 2^{78}, 5)$ for $q = 2$ and $(11, 3^6, 5)$ for $q = 3$. It turns out that there do indeed exist linear binary $[23, 12, 7]$ and ternary $[11, 6, 5]$ codes; these are known as Golay codes. But, as we shall soon, it is impossible for linear or nonlinear $(90, 2^{78}, 5)$ codes to exist.

Exercise: Show that the (n, M, d) triples $(23, 2^{12}, 7)$, $(90, 2^{78}, 5)$ for $q = 2$, and $(11, 3^6, 5)$ for $q = 3$ satisfy the sphere-packing bound (1.1).

Remark: In view of Theorem 1.3, a convenient way of finding a binary $[23, 12, 7]$ Golay code is to construct first the *extended Golay* $[24, 12, 8]$ code, which is just the $[23, 12, 7]$ Golay code augmented with a final parity check in the last position (such that the weight of every codeword is even).

The extended binary Golay $[24, 12, 8]$ code C_{24} can be generated by the matrix G_{24} defined by

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

Remark: We can express $G_{24} = [1_{12} \,|\, A]$, where A is a 12×12 *symmetric matrix*; that is, $A^t = A$.

Exercise: Show that $\boldsymbol{u}\cdot\boldsymbol{v} = 0$ for all rows $\boldsymbol{u}$ and $\boldsymbol{v}$ of G_{24}. Hint: note that the first row of G is orthogonal to itself. Then establish that $\boldsymbol{u}\cdot\boldsymbol{v} = 0$ when $\boldsymbol{u}$ is the second row and $\boldsymbol{v}$ is any row of G_{24}. Then use the cyclic symmetry of the rows of the matrix A' formed by deleting the first column and first row of A.

Remark: The above exercise establishes that the rows of G_{24} are orthogonal to each other. Noting that the weight of each row of G_{24} is 8, we now make use of the following result.

Definition: A linear code C is *self-orthogonal* if $C \subset C^{\perp}$.

Definition: A linear code C is *self-dual* if $C = C^{\perp}$.

Exercise: Let C be a binary linear code with generator matrix G. If the rows of G are orthogonal to each other and have weights divisible by 4, prove that C is self-orthogonal and that the weight of every codeword in C is a multiple of 4.

Remark: Since $k = 12$ and $n - k = 12$, the linear spaces C_{24} and $C_{24}^{\perp}$ have the same dimension. Hence $C_{24} \subset C_{24}^{\perp}$ implies $C_{24} = C_{24}^{\perp}$. This means that the parity check matrix $H_{24} = [A \,|\, 1_{12}]$ for C_{24} is also a generator matrix for C_{24}!

We are now ready to show that distance of C_{24} is 8 and, consequently, that the binary Golay $[23, 12]$ code generated by the first 23 columns of G_{24} must have minimum distance either 7 or 8. But since the second row of this reduced generator matrix is a codeword of weight 7, we can be sure that the minimum distance is exactly 7.

Theorem 4.1 (Extended Golay $[24, 12]$ code) *The $[24, 12]$ code generated by G_{24} has minimum distance 8.*

Proof: We know that the code generated by G_{24} must have weight divisible by 4. Since both G_{24} and H_{24} are generator matrices for the code, any codeword can be expressed either as a linear combination of the rows of G_{24} or as a linear combination of the rows of H_{24}. We now show that a codeword $\boldsymbol{x} \in C_{24}$ cannot have weight 4. It is not possible for the all of the left-most twelve bits of $\boldsymbol{x}$ to be 0 since $\boldsymbol{x}$ must be some nontrivial linear combination of the rows of G_{24}. Likewise, it is not possible for all of the right-most twelve symbols of $\boldsymbol{x}$ to be 0 since $\boldsymbol{x}$ must be some nontrivial linear combination of the rows of H_{24}. It is also not possible for only one of the left-most (right-most) twelve bits of $\boldsymbol{x}$ to be 1 since $\boldsymbol{x}$ would then be one of the rows of G_{24} (H_{24}), none of which has weight 4. The only other possibility is that $\boldsymbol{x}$ is the sum of two rows of G_{24}, but it is easily seen (again using the cyclic symmetry of A') that no two rows of G_{24} differ in only four positions. Since the weight of every codeword in C_{24} must be a multiple of 4, we now know that C_{24} must have a minimum distance of at least 8. In fact, since the second row of G_{24} is a codeword of weight 8, we see that the minimum distance of C_{24} is exactly 8.

Exercise: Show that the ternary Golay $[11, 6]$ code generated by the first 11 columns of the generator matrix

$$G_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 2 & 2 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 1 & 0 \end{bmatrix}$$

has minimum distance 5.

Theorem 4.2 (Nonexistence of $(90, 2^{78}, 5)$ codes) *There exist no $(90, 2^{78}, 5)$ codes.*

Proof: Suppose that a binary $(90, 2^{78}, 5)$ code C exists. By Lemma 1.2, without loss of generality we may assume that $\mathbf{0} \in C$. Let Y be the set of vectors in F_2^{90} of weight 3 that begin with two ones. Since there are 88 possible positions for the third one, $|Y| = 88$. From Eq. (1.1), we know that C is perfect, with $d(C) = 5$. Thus each $\boldsymbol{y} \in Y$ is within a distance 2 from a unique codeword $\boldsymbol{x}$. But then from the triangle inequality,

$$2 = d(C) - w(\boldsymbol{y}) \le w(\boldsymbol{x}) - w(\boldsymbol{y}) \le w(\boldsymbol{x} - \boldsymbol{y}) \le 2,$$

from which we see that $w(\boldsymbol{x}) = 5$ and $d(\boldsymbol{x}, \boldsymbol{y}) = w(\boldsymbol{x} - \boldsymbol{y}) = 2$. This means that $\boldsymbol{x}$ must have a one in every position that $\boldsymbol{y}$ does.

Let X be the set of all codewords of weight 5 that begin with two ones. We know that for each $\boldsymbol{y} \in Y$ there is a unique $\boldsymbol{x} \in X$ such that $d(\boldsymbol{x}, \boldsymbol{y}) = 2$. That is, there are exactly $|Y| = 88$ elements in the set $\{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \in X, \boldsymbol{y} \in Y, d(\boldsymbol{x}, \boldsymbol{y}) = 2\}$. But each $\boldsymbol{x} \in X$ contains exactly three ones after the first two positions. Thus, for each $\boldsymbol{x} \in X$ there are precisely three vectors $\boldsymbol{y} \in Y$ such that $d(\boldsymbol{x}, \boldsymbol{y}) = 2$. That is, $3\,|X| = 88$. This is a contradiction, since $|X|$ must be an integer.

Remark: In 1973, Tietävainen, based on work by Van Lint, proved that any nontrivial perfect code over the field F_q^n must either have the parameters $((q^r - 1)/(q-1), q^{n-r}, 3)$ of a Hamming code, the parameters $(23, 2^{12}, 7)$ of the binary Golay code, or the parameters $(11, 3^6, 5)$ of the ternary Golay code.

Chapter 5

Cyclic Codes

Cyclic codes are an important class of linear codes for which the encoding and decoding can be efficiently implemented using *shift registers.* In the binary case, shift registers are built out of two-state storage elements known as *flip-flops* and arithmetic devices called *binary adders* that output the sum of their two binary inputs, modulo 2.

Many common linear codes, including Hamming and Golay codes, have an equivalent cyclic representation.

Definition: A linear code C is *cyclic* if

$$a_0a_1\dots a_{n-1} \in C \Rightarrow a_{n-1}a_0a_1\dots a_{n-2} \in C.$$

Remark: If $\boldsymbol{x}$ is a codeword of a cyclic code C, then all cyclic shifts of $\boldsymbol{x}$ also belong to C.

- The binary linear code $(000, 101, 011, 110)$ is cyclic.
- The $(7, 16, 3)$ perfect code in Chapter 1, which we now know is equivalent to $\text{Ham}(3, 2)$, is cyclic.
- The binary linear code $(0000, 1001, 0110, 1111)$ is not cyclic. However, upon interchanging the third and fourth positions, we note that it is equivalent to the linear code $(0000, 1010, 0101, 1111)$, which is cyclic.

It is convenient to identify a codeword $a_0a_1\dots a_{n-1}$ in a cyclic code C with the polynomial

$$c(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}.$$

Then $a_{n-1}a_0a_1\dots a_{n-2}$ corresponds to the polynomial

$$a_{n-1} + a_0x + a_1x^2 + \dots + a_{n-2}x^{n-1} = xc(x) \ (\operatorname{mod} x^n - 1),$$

since $x^n = 1 \pmod{x^n - 1}$. Thus, a linear code C is cyclic iff

$$c(x) \in C \Rightarrow xc(x) \pmod{x^n - 1} \in C.$$

That is, multiplication by x (modulo the polynomial $x^n - 1$) corresponds to a cyclic shift.

Definition: The *polynomial ring* $F_q[x]$ is the set of all polynomials $P(x)$ with coefficients in F_q.

Definition: The *residue class ring* $R_n \doteq F_q[x]/(x^n - 1)$ is the set of all polynomial remainders obtained by long division of polynomials in $F_q[x]$ by $x^n - 1$. That is, R_n is the set of all polynomials of degree less than n.

Remark: A cyclic code in F_q^n can be thought of as a particular subset of the residue class polynomial ring R_n. In fact, the following theorem shows that a cyclic code C is an *ideal* of R_n.

Theorem 5.1 (Cyclic Codes are Ideals) *A linear code C in R_n is cyclic* $\iff$

$$c(x) \in C, r(x) \in R_n \Rightarrow r(x)c(x) \in C.$$

Proof: Suppose C is a cyclic code in R_n. We know that multiplication of a codeword $c(x)$ in C by x corresponds to a cyclic shift of its coefficients, and since C is linear, we know that $c(x) \in C \Rightarrow \alpha c(x) \in C$ for all $\alpha \in F_q$. We thus see by induction that

$$c(x) \in C \Rightarrow r(x)c(x) \in C \quad \forall r(x) \in R_n, \tag{5.1}$$

where the multiplication is performed modulo $x^n - 1$. Conversely, suppose that C satisfies Eq. (5.1). Taking $r(x) = x$ shows that C is cyclic.

Definition: The *principal ideal*

$$\langle g(x) \rangle = \{r(x)g(x) : r(x) \in R_n\}$$

of R_n is the cyclic code *generated* by the polynomial $g(x)$.

Exercise: Verify that $\langle g(x) \rangle$ is an ideal.

Remark: The next theorem states that every ideal in R_n is a principal ideal (i.e. R_n is a *Principal Ideal Domain*).

Definition: A polynomial is *monic* if its highest-degree coefficient is 1.

Theorem 5.2 (Generator Polynomial) *Let C be a nonzero q-ary cyclic code in R_n. Then* *(i) there exists a unique monic polynomial $g(x)$ of smallest degree in C;*

(ii) $C = \langle g(x) \rangle$;

(iii) $g(x)$ is a factor of $x^n - 1$ in $F_q[x]$.

Proof:

(i) If $g(x)$ and $h(x)$ are both monic polynomials in C of smallest degree, then $g(x) - h(x)$ is a polynomial in C of smaller degree. Then $g(x) - h(x) \neq 0$ would imply that a certain scalar multiple of $g(x) - h(x)$ is a monic polynomial in C of degree smaller than $\deg g$, which is a contradiction. Hence $g(x) = h(x)$.

(ii) Theorem 5.1 shows that $\langle g(x) \rangle \subset C$, so it only remains to show that $C \subset \langle g(x) \rangle$. Suppose $c(x) \in C$. Using long division, we can express $c(x) = q(x)g(x) + r(x)$, where $\deg r < \deg g$. But since $c(x)$ and $q(x)g(x)$ are both in the cyclic code C, we know by the linearity of C that $r(x) = c(x) - q(x)g(x)$ is also in C. Hence $r(x) = 0$ (otherwise a scalar multiple of $r(x)$ would be a monic polynomial in C of degree smaller than $\deg g$). That is $c(x) \in \langle g(x) \rangle$.

(iii) By long division, we may express $x^n - 1 = q(x)g(x) + r(x)$, where $\deg r < \deg g$. But then $r(x) = -q(x)g(x) \pmod{x^n - 1}$ implies that $r(x) \in \langle g(x) \rangle$. By the minimality of $\deg g$, we see that $r(x) = 0$; that is, $x^n - 1$ is a multiple of $g(x)$.

Definition: The monic polynomial of least degree in Theorem 5.2 is called the *generator polynomial* of C.

Theorem 5.3 (Lowest Generator Polynomial Coefficient) *Let $g(x) = g_0 + g_1 x + \ldots + g_r x^r$ be the generator polynomial of a cyclic code. Then $g_0 \neq 0$.*

Proof: Suppose $g_0 = 0$. Then $x^{n-1} g(x) = x^{-1} g(x)$ is a codeword of C of degree $r - 1$, contradicting the minimality of $\deg g$.

Theorem 5.4 (Cyclic Generator Matrix) *A cyclic code with generator polynomial*

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r$$

has dimension $n - r$ and generator matrix

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & \ldots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & g_0 & g_1 & g_2 & \ldots & g_r \end{bmatrix}.$$

Proof: Let $c(x)$ be a codeword in a cyclic code C with generator $g(x)$. From Theorem 5.2, we know that

$$c(x) = q(x)g(x)$$

for some polynomial $q(x)$. Note that $\deg q < n - r$ since $\deg f < n$. That is,

$$c(x) = \left(q_0 + q_1x + \ldots + q_{n-r-1}x^{n-r-1}\right) g(x) = q_0g(x)+q_1xg(x)+\ldots+q_{n-r-1}x^{n-r-1}g(x),$$

which is a linear combination of the $n - r$ rows $g(x)$, $xg(x)$, $x^2g(x), \ldots, x^{n-r-1}g(x)$ of G. The diagonal of nonzero g_0s next to a lower-triangular zero submatrix ensures that the rows of G are linearly independent. Thus, the span of the rows of G is the $n - r$ dimensional code C.

Remark: Together, Theorems 5.1 and 5.4 say that an $[n, k]$ code is cyclic $\iff$ it is generated by a factor of $x^n - 1$. The following lemma is useful in finding these factors.

Lemma 5.1 (Linear Factors) *A polynomial $c(x)$ has a linear factor $x - a$ $\iff$ $c(a) = 0$.*

Proof: Exercise.

Definition: A polynomial is said to be *irreducible* in $F_q[x]$ if it cannot be factored into polynomials of smaller degree.

Lemma 5.2 (Irreducible 2nd or 3rd Degree Polynomials) *A polynomial $c(x)$ in $F_q[x]$ of degree 2 or 3 is irreducible $\iff$ $c(a) \neq 0$ for all $a \in F_q$.*

Proof: If $c(x)$ can be factored into polynomials of smaller degree $\iff$ it has at least one linear factor $(x - a)$ $\iff$ $c(a) = 0$, by Lemma 5.1.

- Suppose we wish to find all ternary cyclic codes of length $n = 4$. The generators for such codes must be factors of $x^4 - 1$ in the ring $F_3[x]$. Since 1 is a root of the equation $x^4 - 1$ we know that $(x - 1)$ is a factor and hence

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

By Lemma 5.2, the factor $x^3 + x^2 + x + 1$ is not irreducible because it has a linear root at $a = 2 = -1$ in F_3. Using long division, we obtain

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

Since any combination of these three factors can be used to construct a generator polynomial $g(x)$ for a cyclic code, there are a total of $2^3 = 8$ ternary cyclic codes of length 4, as illustrated in Table 5.1. Upon examining the weights of the rows of the possible generator matrices, we see that the generated codes either have minimum distance less than or equal to 2 or else equal to 4. Hence, it is not possible to have a cyclic code of length 4 and minimum distance 3. In particular, Ham(2, 3), for which $n = (3^2 - 1)/(3 - 1) = 4$, cannot be cyclic. That is, not all Hamming codes have a cyclic representation.

$g(x)$	G
1	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$x-1$	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$x+1$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
x^2+1	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
$(x-1)(x+1) = x^2-1$	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
$(x-1)(x^2+1) = x^3-x^2+x-1$	$\begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}$
$(x+1)(x^2+1) = x^3+x^2+x+1$	$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$
$x^4-1 = 0$	$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$

Table 5.1: Generator polynomial $g(x)$ and corresponding generator matrix G for all possible ternary cyclic codes of length 4.

An easy way to find the parity check matrix for a cyclic $[n,k]$ code (without requiring that we first put G given by Theorem 5.4 in standard form) is to first construct the *check polynomial* $h(x)$ of C from its generator polynomial $g(x)$, where $h(x)$ satisfies

$$x^n - 1 = g(x)h(x).$$

Since g is monic and has degree $n-k$, we see that h is monic and has degree k.

Theorem 5.5 (Cyclic Check Polynomial) *An element $c(x)$ of R_n is a codeword of the cyclic code with check polynomial $h \iff c(x)h(x) = 0$ in R_n.*

Proof:

"$\Rightarrow$" If $c(x)$ is a codeword, then in R_n we have

$$c(x) = a(x)g(x) \Rightarrow c(x)h(x) = a(x)g(x)h(x) = a(x)(x^n-1) = a(x)0 \pmod{x^n-1} = 0.$$

"$\Leftarrow$" We can express any polynomial $c(x)$ in R_n as $c(x) = q(x)g(x) + r(x)$ where $\deg r < \deg g = n-k$. If $c(x)h(x) = 0$ then

$$r(x)h(x) = c(x)h(x) - q(x)g(x)h(x) = 0 \pmod{x^n - 1}.$$

But $\deg r(x)h(x) < n-k+k = n$, so $r(x)h(x) = 0$ in $F_q[x]$, not just in R_n. If $r(x) \neq 0$, consider its highest degree coefficient $a \neq 0$. Then since h is monic, the coefficient of the highest degree term of the product $r(x)h(x)$ is $a = a(1) = 0$, which is a contradiction. Thus $r(x) = 0$ and so $c(x)$ is a codeword: $c(x) = q(x)g(x) \in \langle g(x) \rangle$.

Theorem 5.6 (Cyclic Parity Check Matrix) *A cyclic code with check polynomial $h(x)$*

$$h(x) = h_0 + h_1 x + \ldots + h_k x^k$$

has dimension k and parity check matrix

$$H = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \ldots & h_0 & 0 & 0 & \ldots & 0 \\ 0 & h_k & h_{k-1} & h_{k-2} & \ldots & h_0 & 0 & \ldots & 0 \\ 0 & 0 & h_k & h_{k-1} & h_{k-2} & \ldots & h_0 & \ldots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & h_k & h_{k-1} & h_{k-2} & \ldots & h_0 \end{bmatrix}.$$

Proof: Since the degree of the generator polynomial g is $r = n-k$, by Theorem 5.4, the dimension of the code must be k. From Theorem 5.5, we know that a codeword $c(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1}$ must satisfy $c(x)h(x) = 0$. In particular, the coefficients $x^k, x^{k+1}, \ldots, x^{n-1}$ of the product $c(x)h(x)$ must be zero; for $\ell = k, k+1, \ldots, n-1$ we then have

$$0 = \sum_{i+j=\ell} c_i h_j.$$

But then, since each of these equations is one of the $n-k$ rows of the matrix equation

$$\begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \dots & h_0 & 0 & 0 & \dots & 0 \\ 0 & h_k & h_{k-1} & h_{k-2} & \dots & h_0 & 0 & \dots & 0 \\ 0 & 0 & h_k & h_{k-1} & h_{k-2} & \dots & h_0 & \dots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 & h_k & h_{k-1} & h_{k-2} & \dots & h_0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

the codewords are orthogonal to all cyclic shifts of the vector $h_k h_{k-1} h_{k-2} \dots h_0 00 \dots 0$. Hence, the codewords are orthogonal to all linear combinations of the rows of H. This means that $C^\perp$ contains the span of the rows of H. But $h_k = 1$, so we see that H has rank $n-k$ and hence generates exactly the linear subspace $C^\perp$. That is, H is a parity check matrix for the code with check polynomial $h(x)$.

Definition: The *reciprocal polynomial* $\bar{h}(x)$ of a polynomial

$$h(x) = h_0 + h_1 x + \dots + h_k x^k$$

is obtained by reversing the order of its coefficients:

$$\bar{h}(x) \doteq x^k h(x^{-1}) = h_0 x^k + h_1 x^{k-1} + \dots + h_k = h_k + h_{k-1} x + \dots + h_0 x^k.$$

Remark: Since

$$x^{n-k} \bar{h}(x) g(x^{-1}) = x^n h(x^{-1}) g(x^{-1}) = x^n [(x^{-1})^n - 1] = 1 - x^n,$$

we see that $\bar{h}(x)$ is a factor of $x^n - 1$. In view of Theorem 5.2, this says that $C_\perp$ is itself a cyclic code, with (monic) generator $h_0^{-1} \bar{h}(x)$.

We are now able to show that all binary Hamming codes have an equivalent cyclic representation.

Theorem 5.7 (Cyclic Binary Hamming Codes) *The binary Hamming code* $\mathrm{Ham}(r, 2)$ *is equivalent to a cyclic code.*

Proof: Let $p(x)$ be an irreducible polynomial of degree r in $F_2[x]$. By Theorem A.3 $F_2[x]/p(x)$ is a field of order 2^r, and by Theorem A.4 we know that $F_2[x]/p(x)$ can be expressed as the set of distinct elements $\{0, \alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{2^r-2}\}$ for some *primitive element* α. We associate each element $a_0 + a_1 x + a_2 x^2 + \dots + a_{r-1} x^{r-1} \in F_2[x]/p(x)$ with the column vector

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{r-1} \end{bmatrix}.$$

Let $n = 2^r - 1$. The $r \times n$ matrix

$$H = [\,1 \quad \alpha \quad \alpha^2 \quad \ldots \quad \alpha^{n-1}\,]$$

is then seen to be the parity check matrix for $C = \mathrm{Ham}(r, 2)$ since its columns are precisely the distinct nonzero vectors of F_{2^r}. A codeword $c(x) = c_0 + c_1 x + \ldots c_{n-1} x^{n-1}$ in this code must then satisfy the vector equation $c_0 + c_1\alpha_1 + c_1\alpha_2 \ldots + c_{n-1}\alpha^{n-1} = 0$, so that

$$C = \{c(x) \in R_n : c(\alpha) = 0 \text{ in } F_2[x]/p(x)\}$$

If $c(x) \in C$ and $r(x) \in R_n$, we have $r(\alpha)c(\alpha) = r(\alpha)0 = 0$ in $F_2[x]/p(x)$, so $r(x)c(x)$ is also an element of C. Theorem 5.1 then implies that C is cyclic.

- The irreducible polynomial $x^3 + x + 1$ in $F_2[x]$ can be used to generate the field $F_8 = F_2[x]/(x^3 + x + 1)$ with $2^3 = 8$ elements. Note that F_8 has x as a primitive element since all polynomials in $F_2[x]$ of degree less than 3 can be expressed as powers of x:

  $$F_8 = \{0, 1, x, x^2, x^3 = x + 1, x^4 = x^2 + x, x^5 = x^2 + x + 1, x^6 = x^2 + 1\}.$$

  Note that $x^7 = x^3 + x = 1$; that is, the primitive element has order $7 = 8 - 1$. The primitive element x is a root of the *primitive polynomial* $x^3 + x + 1$ in F_8.

- A parity check matrix for a cyclic version of the Hamming code $\mathrm{Ham}(3, 2)$ is thus

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

Q. What is the generator polynomial for $\mathrm{Ham}(r, 2)$?

A. The close parallel between Theorem 5.2 and Theorem A.5 when $n = p^r - 1$ gives us a clue: on comparing these results, we see from Theorem 5.4 that any *minimal polynomial* of F_q is a generator polynomial for a cyclic code in F_q. (see Appendix A). In particular, the following Corollary to Theorem 5.7 establishes that $\mathrm{Ham}(r, 2)$ can be generated by any *primitive polynomial* of F_{2^r}, which is just an irreducible polynomial in $F_2[x]$ having a primitive element as a root.

Corollary 5.7.1 (Binary Hamming Generator Polynomials) *Any primitive polynomial of F_{2^r} is a generator polynomial for a cyclic Hamming code* $\mathrm{Ham}(r, 2)$.

Proof: Let α be a primitive element of F_{2^r}. Its minimal polynomial $p(x)$ is a primitive polynomial of F_{2^r}. From the proof of Theorem 5.7, we see that $\mathrm{Ham}(r, 2)$ consists precisely of those polynomials $c(x)$ for which $c(\alpha) = 0$, for example, $p(x)$ itself. By Theorem A.5, any such polynomial must be a multiple of $p(x)$. That is, $\mathrm{Ham}(r, 2) \subset \langle p(x)\rangle$. Moreover, Theorem 5.1 implies that every multiple of $p(x)$ belongs to the cyclic code $\mathrm{Ham}(r, 2)$. Hence $\mathrm{Ham}(r, 2) = \langle p(x)\rangle$.

- Consider the irreducible polynomial $p(x) = x^3+x+1$ in $F_2[x]$. Since x is a primitive element of $F_2[x]/p(x)$ and $p(x) = 0 \mod(x^3 + x + 1)$, we know that $p(x)$ is a primitive polynomial of $F_{2^3} = F_2[x]/p(x)$ and hence $\text{Ham}(3, 2) = \langle p(x) \rangle$. From Theorem 5.4, we can then immediately write down a generator matrix for a cyclic $\text{Ham}(3, 2)$ code:

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

Chapter 6

BCH Codes

For noisy transmission lines, Hamming codes are of limited use because they cannot correct more than one error. In this chapter, we discuss a class of important and widely used cyclic codes that can correct multiple errors, developed by R. C. Bose and D. K. Ray-Chaudhuri (1960) and independently by A. Hocquenghem (1959), known as *Bose–Chaudhuri–Hocquenghem (BCH) codes.*

Definition: Let α be an element of order n in a finite field F_{q^s}. An $[n,k]$ *BCH code* of *design distance* d is a cyclic code of length n generated by a polynomial $g(x)$ in $F_q[x]$ of degree $n-k$ that has roots at $\alpha, \alpha^2, \ldots, \alpha^{d-1}$.

Remark: Often we take α to be a primitive element of F_{q^s}, so that $n = q^s - 1$. The resulting BCH code is known as a *primitive BCH code.* However, it is possible to construct BCH codes over F_{q^s} of length n, where n is any factor of $q^s - 1$.

Remark: We will establish that a BCH code of odd design distance d has minimum distance of at least d, by showing that such a code can correct $(d-1)/2$ errors.

Exercise: Show that a polynomial $c(x)$ belongs to an $[n,k]$ BCH code of design distance $d \iff c(\alpha) = c(\alpha^2) = \ldots = c(\alpha^{d-1}) = 0$.

To encode the message word $a_0a_1 \ldots a_{k-1}$, we represent it by the polynomial $f(x) = \sum_{i=0}^{k-1} a_i x^i$ and form its product with the generator polynomial $g(x)$, to obtain the codeword $c(x) = f(x)g(x)$.

- For the primitive element $\alpha = x$ of the field $F_{2^4} = F_2[x]/(x^4 + x + 1)$, we can construct a $[15, 7]$ code that can correct two errors, by finding a generator polynomial $g(x)$ that that has roots at α, α^2, a^3, and α^4. Such a generator can be created from the product of the minimal polynomials $m_1(x) = x^4 + x + 1$ of α and $m_3(x) = x^4 + x^3 + x^2 + x + 1$ of α^3:

$$g(x) = m_1(x)m_3(x) = (x^4+x+1)(x^4+x^3+x^2+x+1) = x^8+x^7+x^6+x^4+1.$$

Note that $g(x)$ has even more roots than prescribed, namely at α, α^2, α^4, α^8, α^3, α^6, α^{12}, and α^9. Once we have shown that this code can correct two errors, we will know that its minimum distance is exactly 5 since the codeword $g(x)$ has weight 5.

Remark: In the binary case $q = 2$, we may choose $g(x)$ to be the product of the minimal polynomials of the odd powers, from 1 to $d-1$, of the primitive element.

We now describe the decoding procedure for BCH codes. To keep the notation simple we begin by illustrating the procedure first for the binary case, where $q = 2$. Suppose that $v(x)$ is received rather than $c(x)$ and that t errors have occurred. Then the *error polynomial* $e(x) = v(x) - c(x)$ can be written as $e(x) = x^{\ell_1} + x^{\ell_2} + \ldots + x^{\ell_t}$ for some unknown powers ℓ_1, ℓ_2, $\ldots$, ℓ_t. We then compute the *syndrome* S_1 by substituting α into $v(x)$,

$$S_1 \doteq v(\alpha) = c(\alpha) + e(\alpha) = e(\alpha) = e_1 + \ldots + e_t,$$

where $e_i \doteq \alpha^{\ell_i}$ for $i = 1, 2, \ldots, t$. Likewise, we evaluate

$$S_2 \doteq v(\alpha^2) = c(\alpha^2) + e(\alpha^2) = e(\alpha^2) = e_1^2 + \ldots + e_t^2,$$

$$S_3 \doteq v(\alpha^3) = c(\alpha^3) + e(\alpha^3) = e(\alpha^3) = e_1^3 + \ldots + e_t^3,$$

$$\ldots$$

$$S_{d-1} \doteq v(\alpha^{d-1}) = c(\alpha^{d-1}) + e(\alpha^{d-1}) = e(\alpha^{d-1}) = e_1^{d-1} + \ldots + e_t^{d-1}.$$

The decoding problem now amounts to determining which value of t and which choices of the field elements e_1, e_2, $\ldots$, e_t are consistent with the above equations. Once these are found, from a table of the elements of F_{q^s}, we can determine the corresponding powers ℓ_1, ℓ_2, $\ldots$, ℓ_t such that $e_i \doteq \alpha^{\ell_i}$. These powers tell us directly which bits we need to toggle. To find a solution to the above equations, the following definition will be helpful.

Definition: The *error locator polynomial* is

$$\sigma(x) \doteq (e_1 x - 1)(e_2 x - 1) \ldots (e_t x - 1) = b_t x^t + b_{t-1} x^{t-1} + \ldots + b_1 x + 1.$$

Notice that the roots of $\sigma(x)$ are just the inverses of e_i, $i = 1, 2, \ldots, t$.

To understand how the above syndrome equations are solved, it will be helpful to first discuss the case where $d = 5$ and $t \leq 2$ errors have occurred. We define $e_i = 0$ for $i > t$ and write

$$S_1 = e_1 + e_2,$$

$$S_2 = e_1^2 + e_2^2,$$

$$S_3 = e_1^3 + e_2^3,$$
$$S_4 = e_1^4 + e_2^4.$$

The error locator polynomial is $\sigma(x) = b_2x^2+b_1x+1$. Since $\sigma(e_i^{-1}) = 0$ for $i = 1, 2$ we know that

$$0 = e_1^3\,\sigma(e_1^{-1}) = e_1^3(b_2e_1^{-2} + b_1e_1^{-1} + 1) = b_2e_1 + b_1e_1^2 + e_1^3$$

and

$$0 = e_2^3\,\sigma(e_2^{-1}) = b_2e_2 + b_1e_2^2 + e_2^3.$$

Upon adding these equations, we obtain

$$0 = b_2(e_1 + e_2) + b_1(e_1^2 + e_2^2) + (e_1^3 + e_2^3),$$

i.e.

$$S_1b_2 + S_2b_1 = S_3.$$

If for each i we had multiplied $\sigma(e_i^{-1}) = 0$ by e_i^4 instead of e_i^3 and added the resulting equations, we would have obtained

$$S_2b_2 + S_3b_1 = S_4.$$

To find b_1 and b_2, we only need to solve the system of equations

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} b_2 \\ b_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix} \tag{6.1}$$

If the coefficient matrix in Eq. (6.1) has rank 0, then $S_1 = S_2 = S_3 = S_4 = 0$ and hence $e_1 = e_2$. This would imply that $\ell_1 = \ell_2$, in which case $e(x) = 0$; that is, no error has occurred. That is, the system of equations will have rank 0 $\iff$ no errors have occurred.

Suppose that the coefficient matrix has rank 1. Since $q = 2$, we know that $S_2 = S_1^2$. Note that $S_1 \neq 0$, for otherwise the first equation would imply that $S_3 = 0$ and the rank of the coefficient matrix would be 0. Since the determinant $S_1S_3 - S_2^2 = 0$, we deduce $S_3 = S_1^3$. But

$$e_1^3 + e_2^3 = (e_1 + e_2)^3 \Rightarrow 0 = 3e_1e_2(e_1 + e_2) = 3e_1e_2S_1.$$

This implies that $e_2 = 0$ (only one error has occurred) and that $S_1 = e_1 = \alpha^{\ell_1}$. Conversely, if only one error has occurred, then $S_3 = S_1^3 \neq 0$ and the coefficient matrix of Eq. (6.1) will have rank 1. Using a power table for F_{2^r}, we simply look up the exponent ℓ_1 such that $\alpha^{\ell_1} = S_1$ and then toggle bit ℓ_1 of $v(x)$ to obtain the transmitted codeword $c(x)$.

Finally, if the rank of the coefficient matrix is 2, we can solve for the coefficients b_1 and b_2. If two errors have occurred, the error locator polynomial $\sigma(x) = b_2x^2+b_1x+1$ must have two roots in F_{2^4}, which we can determine by trial and error. The powers of α associated with the inverses of these roots identify the two bit positions in which errors have occurred.

- Let us demonstrate this decoding scheme for the $[15,7]$ BCH code generated by the polynomial $g(x) = x^8 + x^7 + x^6 + x^4 + 1$. Given the message word 110 0000, the transmitted codeword is 110 011 100 100 000, i.e. $c(x) = (1 + x)\, g(x) = x^9 + x^6 + x^5 + x^4 + x + 1$. Suppose that two errors have occurred, so that the received vector is 110 010 101 100 000, that is, $v(x) = x^9 + x^8 + x^6 + x^4 + x + 1$.

  Consulting the power table for $F_2[x]/(x^4 + x + 1)$, we see that the syndromes are

$$\begin{aligned}
S_1 &= v(\alpha) = \alpha^9 + \alpha^8 + \alpha^6 + \alpha^4 + \alpha + 1 \\
&= (\alpha^3 + \alpha) + (\alpha^2 + 1) + (\alpha^3 + \alpha^2) + (\alpha + 1) + \alpha + 1 = \alpha + 1 = \alpha^4, \\
S_2 &= S_1^2 = \alpha^8, \\
S_3 &= v(\alpha^3) = \alpha^{27} + \alpha^{24} + \alpha^{18} + \alpha^{12} + \alpha^3 + 1 \\
&= \alpha^{12} + \alpha^9 + \alpha^3 + \alpha^{12} + \alpha^3 + 1 = \alpha^9 + 1 = \alpha^3 + \alpha + 1 = \alpha^7, \\
S_4 &= S_1^4 = S_2^2 = \alpha^{16} = \alpha.
\end{aligned}$$

  Since $S_1 \neq 0$ and $S_3 - S_1^3 = \alpha^7 - \alpha^{12} \neq 0$, the system of equations,

$$S_1 b_2 + S_2 b_1 = S_3 \Rightarrow \alpha^4 b_2 + \alpha^8 b_1 = \alpha^7,$$

$$S_2 b_2 + S_3 b_1 = S_4 \Rightarrow \alpha^8 b_2 + \alpha^7 b_1 = \alpha,$$

  has rank 2. Upon adding the first equation times α^4 to the second equation, we see that

$$(\alpha^{12} + \alpha^7) b_1 = \alpha^{11} + \alpha.$$

  Thus, $b_1 = \alpha^{-2}\alpha^6 = \alpha^4$ and hence $b_2 = \alpha^{-4}(\alpha^7 + \alpha^8\alpha^4) = \alpha^{-4}\alpha^2 = \alpha^{13}$. Thus, the error polynomial is

$$\sigma(x) = \alpha^{13}x^2 + \alpha^4 x + 1.$$

  We determine the roots of this equation by trial and error, That is, we search through the field until we find an i such that $\alpha^{13+2i} + \alpha^{4+i} = 1$. Incrementing from $i = 0$, the first i we find is $i = 7$, so one root is α^7. If we divide $\sigma(x)$ by $x + \alpha^7$, we obtain the quotient $\alpha^{13}x + \alpha^8$, so the other root is $x = \alpha^{8-13} = \alpha^{10}$. The errors e_1 and e_2 are just the inverses of these roots, namely α^8 and α^5. That is, the two errors are in the fifth and eighth positions, so we decode the received vector 110 010 101 100 000 as the codeword 110 011 100 100 000. Upon division of the associated polynomial $x^9+x^6+x^5+x^4+x+1$ by $g(x) = x^8+x^7+x^6+x^4+1$ we obtain $x + 1$, which corresponds to the original message, 110 0000.

- In the previous example, if instead the received vector was 110 010 100 100 000, that is, $v(x) = x^9 + x^6 + x^4 + x + 1$, the syndromes would be

$$\begin{aligned}
S_1 &= v(\alpha) = \alpha^9 + \alpha^6 + \alpha^4 + \alpha + 1 \\
&= (\alpha^3 + \alpha) + (\alpha^3 + \alpha^2) + (\alpha + 1) + \alpha + 1 = \alpha^2 + \alpha = \alpha^5, \\
S_2 &= S_1^2 = \alpha^{10}, \\
S_3 &= v(\alpha^3) = \alpha^{27} + \alpha^{18} + \alpha^{12} + \alpha^3 + 1 \\
&= \alpha^{12} + \alpha^3 + \alpha^{12} + \alpha^3 + 1 = 1, \\
S_4 &= S_1^4 = S_2^2 = \alpha^{20} = \alpha^5.
\end{aligned}$$

Since $S_3 = S_1^3 \neq 0$, we know that only one error has occurred. In fact, $S_1 = \alpha^5$, so the error must be in the fifth position; one again we see that the transmitted codeword was 110 011 100 100 000.

In general, decoding requires that we solve the **nonlinear** system of $d-1$ syndrome equations

$$S_i = v(\alpha^i) = e_1^i + \ldots + e_t^i, \qquad i = 1, \ldots, d-1 \tag{6.2}$$

for the values e_i and t. Here $t \leq (d-1)/2$ is the actual number of errors that have occurred, so that each of the values e_j for $j = 1, \ldots, t$ are nonzero and distinct.

A straightforward generalization of the $t = 2$ decoding scheme leads to the t equations:

$$\begin{aligned}
0 = e_i^{t+1}\,\sigma(e_i^{-1}) &= b_t e_i + b_{t-1} e_i^2 + \ldots + b_1 e_i^t + e_i^{t+1}, \\
0 = e_i^{t+2}\,\sigma(e_i^{-1}) &= b_t e_i^2 + b_{t-1} e_i^3 + \ldots + b_1 e_i^{t+1} + e_i^{t+2}, \\
&\ldots \\
0 = e_i^{2t}\,\sigma(e_i^{-1}) &= b_t e_i^t + b_{t-1} e_i^{t+1} + \ldots + b_1 e_i^{2t-1} + e_i^{2t}.
\end{aligned}$$

On summing each of these equations over i, we obtain a **linear** system of equations for the values $b_1, b_2, \ldots, b_t$ in terms of the $2t \leq d-1$ syndromes $S_1, S_2, \ldots, S_{2t}$:

$$\begin{bmatrix} S_1 & S_2 & \ldots & S_t \\ S_2 & S_3 & \ldots & S_{t+1} \\ \vdots & \vdots & & \vdots \\ S_t & S_{t+1} & \ldots & S_{2t-1} \end{bmatrix} \begin{bmatrix} b_t \\ b_{t-1} \\ \vdots \\ b_1 \end{bmatrix} = \begin{bmatrix} S_{t+1} \\ S_{t+2} \\ \vdots \\ S_{2t} \end{bmatrix}. \tag{6.3}$$

Exercise: Show that we may rewrite the coefficient matrix in Eq. (6.3) as

$$M = VDV^t,$$

where V is the *Vandermonde* matrix

$$V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ e_1 & e_2 & \ldots & e_t \\ e_1^2 & e_2^2 & \ldots & e_t^2 \\ \vdots & \vdots & & \vdots \\ e_1^{t-1} & e_2^{t-1} & \ldots & e_t^{t-1} \end{bmatrix}$$

and $D = \operatorname{diag}(e_1, e_2, \ldots, e_t)$ is a diagonal matrix with components $d_{ii} = e_i$.

Remark: The matrix D is nonsingular $\iff$ each of its eigenvalues $e_j, j = 1, \ldots, t$ are nonzero. Also, the following theorem establishes that the matrix V is nonsingular $\iff$ the values e_j are distinct.

Theorem 6.1 (Vandermonde Determinants) *For $t \geq 2$ the $t{\times}t$ Vandermonde matrix*

$$V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ e_1 & e_2 & \ldots & e_t \\ e_1^2 & e_2^2 & \ldots & e_t^2 \\ \vdots & \vdots & & \vdots \\ e_1^{t-1} & e_2^{t-1} & \ldots & e_t^{t-1} \end{bmatrix}$$

has determinant $\displaystyle\prod_{\substack{i,j=1\\ i>j}}^{t} (e_i - e_j)$.

Proof: When $t = 2$ we see that $V = e_2 - e_1$. For $t > 2$, suppose that all $(t-1){\times}(t-1)$ Vandermonde matrices have determinant

$$\prod_{\substack{i,j=1\\ i>j}}^{t-1} (e_i - e_j) = \prod_{i=1}^{t-1}\prod_{j=1}^{i-1}(e_i - e_j).$$

By subtracting e_t times row $i-1$ from row i, for $i = t, t-1, \ldots, 2$, we can rewrite the determinant of any $t{\times}t$ Vandermonde matrix as

$$\begin{vmatrix} 1 & 1 & \ldots & 1 & 1 \\ e_1 - e_t & e_2 - e_t & \ldots & e_{t-1} - e_t & 0 \\ e_1(e_1 - e_t) & e_2(e_2 - e_t) & \ldots & e_{t-1}(e_{t-1} - e_t) & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ e_1^{t-2}(e_1 - e_t) & e_2^{t-2}(e_2 - e_t) & \ldots & e_{t-1}^{t-2}(e_{t-1} - e_t) & 0 \end{vmatrix}$$

$$= (-1)^{t-1} \begin{vmatrix} e_1 - e_t & e_2 - e_t & \ldots & e_{t-1} - e_t \\ e_1(e_1 - e_t) & e_2(e_2 - e_t) & \ldots & e_{t-1}(e_{t-1} - e_t) \\ \vdots & \vdots & & \vdots \\ e_1^{t-2}(e_1 - e_t) & e_2^{t-2}(e_2 - e_t) & \ldots & e_{t-1}^{t-2}(e_{t-1} - e_t) \end{vmatrix}$$

$$= \begin{vmatrix} 1 & 1 & \ldots & 1 \\ e_1 & e_2 & \ldots & e_{t-1} \\ e_1^2 & e_2^2 & \ldots & e_{t-1}^2 \\ \vdots & \vdots & & \vdots \\ e_1^{t-2} & e_2^{t-2} & \ldots & e_{t-1}^{t-2} \end{vmatrix} (e_t - e_1)(e_t - e_2)\ldots(e_t - e_{t-1})$$

$$= \prod_{i=1}^{t-1}\prod_{j=1}^{i-1}(e_i - e_j) \cdot \prod_{j=1}^{t-1}(e_t - e_j) = \prod_{i=1}^{t}\prod_{j=1}^{i-1}(e_i - e_j).$$

Remark: We thus see that the matrix $M = VDV^t$ is nonsingular $\iff$ the error values e_j are nonzero and distinct.

Remark: If we attempt to increase the value of t in Eq. (6.2) beyond the actual number of errors that have occurred, either the values e_j will no longer be distinct or at least one of them will be zero. In either case, M will no longer be invertible. This gives us a method for finding the number of errors: t is just the largest number such that

$$M \doteq \begin{bmatrix} S_1 & S_2 & \dots & S_t \\ S_2 & S_3 & \dots & S_{t+1} \\ \vdots & \vdots & & \vdots \\ S_t & S_{t+1} & \dots & S_{2t-1} \end{bmatrix}$$

is invertible.

Remark: If it is *a priori* known that no more than t errors have occurred in a received vector v, then it is impossible for a $(t+1) \times (t+1)$ or larger syndrome matrix based on v to be invertible.

Remark: Once we have determined the maximum value of t such that the coefficient matrix M is invertible, we simply solve the linear system Eq. (6.3) for the coefficients $b_1, b_2, \dots, b_t$ of the error locator polynomial $\sigma(x)$. We can determine all t roots of $\sigma(x)$ simply by searching through all of the field elements (at most one pass is required). The exponents ℓ_1, ℓ_2, ..., ℓ_t corresponding to the inverses of these roots precisely identify the t positions of the received vector that are in error.

Remark: The above decoding procedure can be easily extended to nonbinary codes. In this case, the error vector becomes $e(x) = q_1 x^{\ell_1} + q_2 x^{\ell_2} + \dots + q_t x^{\ell_t}$, where the $q_i \in \{0, 1, \dots, q-1\}$, the syndromes become $S_i = q_1 e_1^i + \dots + q_t e_t^i$, and $D = \text{diag}(q_1 e_1, q_2 e_2, \dots, q_t e_t)$. We then see that any BCH code of design distance d can correct $\lfloor (d-1)/2 \rfloor$ errors. We encapsulate this result in the following theorem.

Theorem 6.2 (BCH Bound) *The minimum distance of a BCH code of odd design distance d is at least d.*

Proof: This follows from Theorem 1.1 and the fact that the BCH code can correct $(d-1)/2$ errors.

Remark: Although Theorem 6.2 may be shown to hold also when the design distance d is even, we are normally interested only in the case of odd d.

Remark: It may happen that the minimum distance of a BCH code exceeds its design distance d, as illustrated by the following example.

- Let α be a primitive element of $F_{2^{11}}$. Since $2^{11} - 1 = 2047 = 23 \times 89$, the element $\beta = \alpha^{89}$ has order $n = 23$. The cyclotomic cosets mod 23 are $\{0\}$, $\{1, 2, 4, 8, 16, 9, 18, 13, 3, 6, 12\}$, and $\{5, 10, 20, 17, 11, 22, 21, 19, 15, 7, 14\}$. Thus the minimal polynomials of $m_1(x)$ of β and $m_5(x)$ of β^5 in $F_{2^{11}}$ each have degree 11.[1] We can then construct a $[23, 12]$ BCH code of length 23 from the degree 11 generator polynomial $m_1(x)$, which has roots at β, β^2, β^3, and β^4. While the design distance of this code is 5, the actual minimum distance is 7; in fact, this BCH code is equivalent to the triple-error correcting $[23, 12, 7]$ Golay code we encountered in Chapter 4.

Remark: The special case of a BCH code where $s = 1$, that is, the primitive element α comes from the same field F_q as the coefficients of the generator polynomial, is known as a *Reed–Solomon* code. Note that the minimal polynomial of any element of F_q has degree $s = 1$. The generator polynomial of a Reed–Solomon code of design distance d,

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3) \dots (x - \alpha^{d-1}),$$

thus has degree $n - k = d - 1$. That is, the minimum distance of the code must at least $n - k + 1$. But since there are $\operatorname{rk} H \le n - k$ independent columns in the parity check matrix, we know from Theorem 2.2 that the minimum distance can be no more than $n - k + 1$. Thus, a Reed–Solomon code achieves the so-called *singleton* upper bound $n - k + 1$ for the minimum distance of a code. Because Reed–Solomon codes are optimal in this sense and easily implementable in hardware (using shift registers), they are widely used for error correction in computer memory chips, magnetic and optical (compact disk) storage systems, high-speed modems, and data transmission channels.

- Since 2 is a primitive element of $\mathbb{Z}_{11}$, the polynomial

  $$g(x) = (x-2)(x-4)(x-8)(x-5)(x-10)(x-9) = x^6+6x^5+5x^4+7x^3+2x^2+8x+2$$

  generates a triple-error correcting $[10, 4, 7]$ Reed–Solomon code over $\mathbb{Z}_{11}$. One of the codewords is $g(x)$ itself, which has weight 7, so the design distance in this case is the actual minimum distance.

- Compact disk players use a double-error correcting $[255, 251, 5]$ Reed–Solomon code over F_{256} (the symbols are eight-bit bytes) with interleaving of the data over the disk to increase robustness to localized data loss.

- High-performance computer memory chips containing so-called ECC SDRAM use Reed–Solomon codes to correct one error per 64-bit word in addition to detecting very rare multiple errors (which are programmed to generate a machine halt).

[1]Since $\beta^{23} = 1$, we know from Theorem A.5 that $x^{23} - 1 = (x - 1)m_1(x)m_5(x)$, moreover, Theorem A.7 implies that $m_5(x) = \bar{m}_1(x)$

Chapter 7

Cryptographic Codes

In contrast to error-correcting codes, which are designed only to increase the reliability of data communications, cryptographic codes are designed to increase their security. In cryptography, the sender uses a key to encrypt a message before it is sent through an insecure channel (such as a telephone line, radio transmission or the internet). An authorized receiver at the other end then uses a key to decrypt the received data to a message. Often, data compression algorithms and error-correcting codes are used in tandem with cryptographic codes to yield communications that are both efficient, robust to data transmission errors, and secure to eavesdropping and/or tampering. Typically, data compression is performed first; the resulting compressed data is then encrypted and finally encoded with an error-correcting scheme before being transmitted through the channel.

Definition: Let $\mathcal{K}$ be a set of cryptographic keys. A *cryptosystem* is a set

$$\{e, d, \mathcal{E}_e, \mathcal{D}_d : e \in \mathcal{K}\}$$

of encrypting and decrypting keys, e and d, and their associated encrypting function $\mathcal{E}_e$ and $\mathcal{D}_d$, respectively.

Most cryptosystems, or *ciphers*, fall into one of two broad classes: *symmetric-key cryptosystems*, where essentially the same key is used both to encrypt and decrypt a message (precisely, where d can be easily determined whenever e is known) and *public-key cryptosystems*, where the encryption key e is made publicly available, but the decryption key d is kept secret and is (hopefully) known only to the receiver.

7.A Symmetric-Key Cryptography

One of the simplest cryptographic system is the *shift cipher* employed by Julius Caeser. Shift ciphers encode each symbol $m \in \{0, 1, \ldots, n-1\}$ in the message as

$$c = \mathcal{E}_e(m) = m + e \pmod{n},$$

where $e \in \mathbb{N}$. Decoding is accomplished *via* the inverse transformation

$$m = \mathcal{D}_d(c) = c + d \ (\operatorname{mod} n),$$

where $d = -e$. That is, encoding is accomplished by addition modulo e and decoding key is accomplished by subtraction modulo e. Caeser adopted the value $e = 3$ to encrypt the $n = 26$ symbols of the Roman alphabet, using 0 to represent the letter A and 25 to represent the letter Z. Some fans of the film "2001: A Space Odyssey" even suggest that the computer name HAL is really a shift cipher for IBM, with $e = 25$!

A slight generalization of the shift cipher is the *affine cipher*, defined by

$$c = \mathcal{E}_{a,b}(m) = am + b \ (\operatorname{mod} n),$$

where $a \in \mathbb{N}$ is relatively prime to n. This condition guarantees the existence of an inverse transformation,

$$m = \mathcal{D}_{a,b}(c) = a^{-1}(c - b) \ (\operatorname{mod} n).$$

Both shift and affine ciphers are very insecure since they are easily decoded simply by trying all possible values of a and b! They are both special cases of *simple substitution ciphers* or *monoalphabetic substitution ciphers*, which permute the alphabet symbol in a prescribed manner. Simple substitution ciphers can be cryptanalyzed (decoded by an unauthorized third party) by *frequency analysis*, in which the encrypted symbol frequencies are compared to those of the original message language to determine the applied permutation. *Block substitution ciphers* or *polyalphabetic substitution ciphers* divide the message into blocks of length r and apply different permutations to the symbols in individual positions of the block. Given enough encrypted text, block substitution ciphers are also easily cryptanalyzed once the block size r is determined, simply by doing a frequency analysis on the letters in each fixed position of all blocks.

Digraph ciphers map pairs of letters in the message text to encrypted pairs of letters. A general example of this is the *linear block* or *Hill cipher*, which uses an $r \times r$ invertible matrix e to encrypt an entire block of r message symbols:

$$c = \mathcal{E}_e(m) = em \ (\operatorname{mod} n),$$

$$m = \mathcal{D}_e(c) = e^{-1}c \ (\operatorname{mod} n).$$

The existence of e^{-1} requires that $\det e$ have an inverse in $\mathbb{Z}_n$, which happens only when $\gcd(\det e, n) = 1$.

- Choose $r = 2$, $n = 26$ and

$$e = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}.$$

We see that $\det e = 5$ has no common factors with 26. To find the inverse of 5 in $\mathbb{Z}_{26}$ we use the Euclidean division algorithm: $1 = 5x + 26y, 26 = 5 \cdot 5 + 1 \Rightarrow 1 = 26 - 5 \cdot 5$, from which we see that $x = -5$ is a solution. Thus

$$e^{-1} = -5 \begin{bmatrix} 4 & -1 \\ -3 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 5 \\ 15 & 16 \end{bmatrix}.$$

We can use e to encrypt the word "SECRET", in other words the message 18 4 2 17 4 19, by breaking the message up into vectors of length two: [18 4], [2 17], [4 19] and then multiplying the transpose of each vector by e on the left. The result is [14 18], [21 22], [1 10], or the cipher text "OSVWBK". Note that the two letters "E" are not mapped to the same symbol in the ciphertext. For this reason the Hill cipher is less susceptible to frequency analysis (particularly for large block sizes; however, the number of entries in the key matrix then becomes unreasonably large).

Exercise: Verify that the original message "SECRET" is recovered when "OSVWBK" is decoded with the matrix e^{-1}.

A special case of the block cipher is the *permutation cipher*, in which the order of the characters in every block of text are rearranged in a prescribed manner. They can be detected by frequency analysis since they preserve the frequency distribution of each symbol. All of the linear or affine block methods are subject to cryptanalysis using linear algebra, once r or $r + 1$ plaintext–ciphertext pairs are known.

A widely used commercial symmetric-key cryptosystem is the *Data Encryption Standard (DES)*, which is a type of *Feistel* cipher endorsed in 1977 by the United States National Bureau of Standards for the protection of confidential (but unclassified) government data. In 1981, DES was approved also for use in the private-sector. Feistel ciphers are block ciphers over F_2^{2t}. One divides a plaintext block message of $2t$ bits into two halves, L_0 and R_0, and then performs the iteration

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} + f(R_{i-1}, e_i), \qquad i = 1, \ldots, r, \end{aligned}$$

where the e_i are the keys and f is a specific nonlinear cipher function. The encryption function is $\mathcal{E}_e(L_0 \,|\, R_0) = R_r \,|\, L_r$, where $|$ denotes concatenation, and $e = (e_1, \ldots e_r)$ denotes a set of r encryption keys. The decryption function $\mathcal{D}_e(L_r \,|\, R_r) = R_0 \,|\, L_0$, is implemented by applying the inverse iteration

$$\begin{aligned} L_{i-1} &= R_i + f(L_{i-1}, e_i), \\ R_{i-1} &= L_i, \qquad i = r, \ldots, 1. \end{aligned}$$

With Feistel ciphers, the encryption and decryption algorithms are essentially the same, except that the key sequence is reversed.

The DES cipher uses the half width $t = 32$ and $r = 16$ rounds of encryption. All 16 keys are generated from a bit sequence of length 64 that are divided into 8 bytes. The eighth bit of each byte is an (odd) parity check, so in fact there are only 56 information bits in the key. Hence the number of keys that need to be searched in a brute-force attack on DES is 2^{56}. In fact, because of an inherent ones-complement symmetry, only 2^{55} keys need to be checked. On a modern supercomputer this is quite feasible; moreover, DES has recently been shown to be susceptible to relatively new cryptanalytic techniques, such as *differential cryptanalysis* and *linear cryptanalysis*. Somewhat more secure variants of DES (such as *Triple-DES*, where DES is applied three times in succession with three different keys), has been developed as an interim solution. One common application of DES that persists today is its use in encrypting computer passwords, using the password itself as a key. This is why many computer passwords are still restricted to a length of eight characters (64 bits).

In October 2000, the *Rijndael Cryptosystem* was adopted by the National Bureau of Standards as the Advanced Encryption Standard (AES) to replace DES. It is based on a combinations of byte substitutions, shifts, and key additions, along with a diffusion-enhancing technique based on cyclic coding theory, where the data values are multiplied by the polynomial $3x^3+x^2+x+2$ in the polynomial ring $R_4 = F_2[x]/(x^4-1)$.

7.B Public-Key Cryptography

A principle difficulty with symmetric-key cryptosystems is the problem of key distribution and management. Somehow, both parties, which may be quite distant from each other, have to securely exchange their secret keys before they can begin communication.

One technique for avoiding the problem of key exchange makes use of two secure *envelopes*, or locks, which each party alternately applies and later removes from the sensitive data, as the data makes a total of three transits between sender and receiver. The required three transmissions makes this method awkward to use in practice.

In public-key cryptosystems, key exchange is avoided altogether by making copies of the receiver's encrypting key (lock) available to anyone who wants to communicate with him. Both the secure envelope technique and the public-key technique require that the encrypting key e is designed so that the decrypting key d is extremely difficult to determine from knowledge of e. They also require authentication of the lock itself, to guard against so-called *man-in-the-middle* attacks.

7.B.1 RSA Cryptosystem

The most well known public-key cipher is the *Rivest–Shamir–Aldeman (RSA) Cryptosystem.* First, the receiver forms the product n of two distinct large primes numbers p and q chosen at random, but such that p and q cannot be easily determined from

n.[1] The receiver then selects a random integer e between 1 and $\varphi(n) = (p-1)(q-1)$ that is relatively prime to $\varphi(n)$ and, using the Euclidean division algorithm, computes $d = e^{-1}$ in $\mathbb{Z}_{\varphi(n)}$ (why does e^{-1} exist?). The numbers n and e are made publicly available, but d, p, q are kept secret.

Anyone who wishes to send a message m, where $0 \le m < n$, to the receiver encrypts the message using the encoding function

$$c = \mathcal{E}_e(m) = m^e \pmod{n}$$

and transmits c. Because the receiver has knowledge of d, the receiver can decrypt c using the decoding function

$$M = \mathcal{D}_e(c) = c^d \pmod{n}.$$

To show that $M = m$, we will need the following results.

Theorem 7.1 (Modified Fermat's Little Theorem) *If s is prime and a and m are natural numbers, then*

$$m\left[m^{a(s-1)} - 1\right] = 0 \pmod{s}.$$

Proof: If m is a multiple of s we are done. Otherwise, we know that m^a is not a multiple of s, so Fermat's Little Theorem[2] implies that $(m^a)^{s-1} = 1 \pmod{s}$, from which the result follows.

Corollary 7.1.1 (RSA Inversion) *The RSA decoding function $\mathcal{D}_e$ is the inverse of the RSA encoding function $\mathcal{E}_e$.*

By construction $ed = 1 + k\varphi(n)$ for some integer k, so

$$M = \mathcal{D}_e(c) = c^d = (m^e)^d = m^{ed} = m^{1+k\varphi(n)} = m^{1+k(p-1)(q-1)} \pmod{n}.$$

We first apply Theorem 7.1 with $a = k(q-1)$, $s = p$ and then with $a = k(p-1)$, $s = q$, to deduce that $m[m^{k(q-1)(p-1)} - 1]$ is a multiple of both of the distinct primes p and q, that is, $m[m^{k(q-1)(p-1)} - 1] = 0 \pmod{pq}$. Thus

$$M = mm^{k(q-1)(p-1)} = m \pmod{pq} = m \pmod{n}.$$

[1]For example, if p and q are close enough that $(p+q)^2 - 4n = (p+q)^2 - 4pq = (p-q)^2$ is small, then the sum $p+q$ could be determined by searching for a small value of $p-q$ such that $(p-q)^2 + 4n$ is a perfect square, which must be $p+q$. Knowledge of $p-q$ and $p+q$ is sufficient to determine both p and q.

[2]This follows from applying Theorem A.4 to the field $\mathbb{Z}_s$.

- Let us encode the message "SECRET" (18 4 2 17 4 19) using the RSA scheme with a block size of 1. The receiver chooses $p = 5$ and $q = 11$, so that $n = pq = 55$ and $\varphi(n) = 40$. He then selects $e = 17$ and finds $d = e^{-1}$ in $\mathbb{Z}_{40}$, so that $17d = 40k + 1$ for some $k \in \mathbb{N}$. This amounts to finding $\gcd(17, 40)$:

$$40 = 2 \cdot 17 + 6, \qquad 17 = 2 \cdot 6 + 5, \qquad 6 = 1 \cdot 5 + 1,$$

from which we see that

$$1 = 6 - 5 = 6 - (17 - 2 \cdot 6) = 3 \cdot (40 - 2 \cdot 17) - 17 = 3 \cdot 40 - 7 \cdot 17.$$

That is, $d = -7 \pmod{40} = 33$. The receiver publishes the numbers $n = 55$ and $e = 17$, but keeps the factors $p = 5$, $q = 11$, and $d = 33$ (and $\varphi(n)$) secret.

The sender then encodes 18 4 2 17 4 19 as

$$18^{17}\ 4^{17}\ 2^{17}\ 17^{17}\ 4^{17}\ 19^{17} \pmod{55} = 28\ 49\ 7\ 52\ 49\ 24$$

The two Es are encoded in exactly the same way, since the block size is 1: obviously, a larger block size should be used to thwart frequency analysis attacks.[3]

The receiver would then decode the received message 28 49 7 52 49 24 as

$$28^{33}\ 49^{33}\ 7^{33}\ 52^{33}\ 49^{33}\ 24^{33} \pmod{55} = 18\ 4\ 2\ 17\ 4\ 19.$$

Remark: While the required exponentiations can be performed by repeated squaring and multiplication in $\mathbb{Z}_{\varphi(n)}$ (e.g. $x^{33} = x^{32} \cdot x$), RSA decryption can be implemented in a more efficient manner. This is important, since to make computing the secret key d (from knowledge of n and e alone) difficult, d must be chosen to be about as large as n. Instead of computing $m = c^d$ directly, we first compute $a = c^d \pmod{p}$ and $b = c^d \pmod{q}$. This is very easy since Fermat's Little Theorem says that $c^{p-1} = 1 \pmod{p}$, so these definitions reduce to

$$a = c^{d \bmod (p-1)} \pmod{p}, \qquad b = c^{d \bmod (q-1)} \pmod{q}.$$

The Chinese Remainder Theorem then guarantees that the system of linear congruences

$$m = a \pmod{p}, \qquad m = b \pmod{q}$$

has exactly one solution in $\{0, 1, \ldots, n-1\}$. One can find this solution by using the Euclidean division algorithm to construct integers x and y such that $1 = xp + yq$. Since $yq = 1 \pmod{p}$ and $xp = 1 \pmod{q}$, we see that

$$m = ayq + bxp \pmod{n}$$

is the desired solution. Since the numbers x and y are independent of the ciphertext the factors px and py can be precomputed.

[3]For example, we could encode pairs of letters i and j as $26i + j$ and choose $n \geq 26^2 = 676$, although such a limited block size would still be vulnerable to more time consuming but feasible digraph frequency attacks

- To set up an efficient decoding scheme we precompute x and y such that $1 = 5x + 11y$. We see that $x = -2$ and $y = 1$ are solutions, so that $xp = -10$ and $yq = 11$. Once a and b are determined from the ciphertext we can quickly compute $m = 11a - 10b \pmod n$. For example, to compute 28^{33} we evaluate

$$a = 28^{33 \pmod 4} \pmod 5 = 28 \pmod 5 = 3,$$

$$b = 28^{33 \pmod{10}} \pmod{11} = 28^3 \bmod 11 = 6^3 \bmod 11 = 7$$

and then compute $m = 11a - 10b = (33 - 70) \pmod{55} = 18$.

Remark: Determining d from e and n can be shown to be equivalent to determining the prime factors p and q of n. Since factoring large integers in general is an extremely difficult problem, the belief by many that RSA is a secure cryptographic system rests on this equivalence. However, it has not been ruled out that no other technique for decrypting RSA ciphertext exists. If such a technique exists, presumably it does not involve direct knowledge of d (as that would constitute an efficient algorithm for factorizing integers!).

7.B.2 Rabin Public-Key Cryptosystem

In contrast to the RSA scheme, the *Rabin Public-Key Cryptosystem* has been proven to be as secure as factorizing large integers is difficult. Again the receiver forms the product $n = pq$ of two large distinct primes p and q that are kept secret. To make decoding efficient, p and q are normally chosen to be both congruent to $3 \pmod 4$. This time, the sender encodes the message $m \in \{0, 1, \ldots, n-1\}$ as

$$c = \mathcal{E}_e(m) = m^2 \pmod n.$$

To decode the message, the receiver must be able to compute square roots modulo n. This can be efficiently accomplished in terms of integers x and y satisfying $1 = xp+yq$. First one notes from Lemma 5.1 that the equation $0 = x^2 - c$ has at most two solutions in $\mathbb{Z}_p$. In fact, these solutions are given by $\pm a$, where $a = c^{(p+1)/4} \pmod p$:

$$(\pm a)^2 = c^{(p+1)/2} \pmod p = cc^{(p-1)/2} \pmod p = cm^{(p-1)} \pmod p = c \pmod p.$$

Similarly, the two square roots of c in $\mathbb{Z}_q$ are $\pm b$, where $b = c^{(q+1)/4} \pmod q$. Consequently, by the Chinese Remainder Theorem, the linear congruences

$$M = \pm a \pmod p, \qquad M = \pm b \pmod q$$

yield four solutions:

$$M = \pm(ayq \pm bxp) \pmod n,$$

one of which is the original message m.

7.B.3 Cryptographic Error-Correcting Codes

We conclude with an interesting cryptographic application of error-correcting codes due to McEliece [1978]. The receiver selects a block size k and a private key consisting of an $[n, k, 2t+1]$ linear code C with generator matrix G, a $k \times k$ nonsingular *scrambler matrix* S, and an $n \times n$ random *permutation matrix* P. He then constructs the $k \times n$ matrix $K = SGP$ as his public key. A sender encodes each message block m as

$$c = \mathcal{E}_e(m) = mK + z,$$

where z is a random error vector of length n and weight no more than t. The receiver then computes

$$cP^{-1} = (mK + z)P^{-1} = (mSGP + z)P^{-1} = mSG + zP^{-1}.$$

Since the weight of zP^{-1} is no more than t, he can use the code C to decode the vector $mSG + zP^{-1}$ to the codeword mS. After multiplication on the right by S^{-1}, he recovers the original message m.

Appendix A

Finite Fields

Theorem A.1 ($\mathbb{Z}_n$) *The ring $\mathbb{Z}_n$ is a field $\iff$ n is prime.*

Proof:

> "$\Rightarrow$" Let $\mathbb{Z}_n$ be a field. If $n = ab$, with $1 < a, b < n$, then $b = a^{-1}ab = 0 \pmod n$, a contradiction. Hence n must be prime.
>
> "$\Leftarrow$" Let n be prime. Since $\mathbb{Z}_n$ has a unit and is commutative, we need only verify that each element $a \neq 0$ has an inverse. Consider the elements ia, for $i = 1, 2, \ldots n-1$. Each of these elements must be nonzero since neither i nor a is divisible by the prime number n. These $n-1$ elements are distinct from each other since, for $i, j \in 1, 2, \ldots n-1$,
>
> $$ia = ja \Rightarrow (i-j)a = 0 \pmod n \Rightarrow n|(i-j)a \Rightarrow n|(i-j) \Rightarrow i = j.$$
>
> Thus, the $n-1$ elements a, $2a$, $\ldots$, $(n-1)a$ must be equal to the $n-1$ elements 1, 2, $\ldots n-1$ in some order. One of them, say ia, must be equal to 1. That is, a has inverse i.

Definition: The *order* of a finite field F is the number of elements in F.

Theorem A.2 (Subfield Isomorphic to $\mathbb{Z}_p$) *Every finite field has the order of a power of a prime p and contains a subfield isomorphic to $\mathbb{Z}_p$.*

Proof: Let 1 (one) denote the (unique) multiplicative identity in F, a field of order n. The element $1+1$ must be in F, so label this element 2. Similarly $2+1 \in F$, which we label by 3. We continue in this manner until the first time we encounter an element k to which we have already assigned a label ℓ (F is a finite field). That is, the sum of k ones must equal to the sum of ℓ ones, where $k > \ell$. Hence the sum of $p \doteq k - \ell$ ones must be the additive identity, 0. If p is composite, $p = ab$, then the product of the elements which we have labelled a and b would be 0, contradicting the fact that F is a field. Thus p must be prime and the set of numbers that we have labelled $\{0, 1, 2, \ldots, p-1\}$ is isomorphic to the field $\mathbb{Z}_p$. Consider all subsets $\{x_1, \ldots, x_r\}$ of linearly independent elements of F, in the sense that

$$a_1x_1 + a_2x_2 + \ldots + a_rx_r = 0 \Rightarrow a_1 = a_2 = \ldots = 0, \text{ where } a_i \in \mathbb{Z}_p.$$

There must be at least one such subset having a maximal number of elements. Then, if x is any element of F, the elements $\{x, x_1, \ldots, x_r\}$ cannot be linearly independent, so that x can be written as a linear combination of $\{x_1, \ldots, x_r\}$. Thus $\{x_1, \ldots, x_r\}$ forms a basis for F, so that the elements of F may be **uniquely** identified by all possible values of the coefficients $a_1, a_2, \ldots, a_r$. Since there are p choices for each of the r coefficients, there are exactly p^r distinct elements in F.

Corollary A.2.1 (Isomorphism to $\mathbb{Z}_p$) *Any field F with prime order p is isomorphic to $\mathbb{Z}_p$.*

Proof: Theorem A.2 says that the prime p must be the power of a prime, which can only be p itself. It also says that F contains $\mathbb{Z}_p$. Since the order of $\mathbb{Z}_p$ is already p, there can be no other elements in F.

Theorem A.3 (Prime Power Fields) *There exists a field F of order $n \iff n$ is a power of a prime.*

Proof:

"$\Rightarrow$" This is implied by Theorem A.2.

"$\Leftarrow$" Let p be prime and g be an irreducible polynomial of degree r in the polynomial ring $\mathbb{Z}_p[x]$ (for a proof of the existence of such a polynomial, see van Lint [1991]). Recall that every polynomial can be written as a polynomial multiple of g plus a residue polynomial of degree less than r. The field $\mathbb{Z}_p[x]/g$, which is just the residue class polynomial ring $\mathbb{Z}_p[x] \pmod g$, establishes the existence of a field with exactly p^r elements, corresponding to the p possible choices for each of the r coefficients of a polynomial of degree less than r.

- For example, we can construct a field with $8 = 2^3$ elements using the polynomial $g(x) = x^3 + x + 1$ in $\mathbb{Z}_2[x]$. Note that g is irreducible, because if $g(x) = (x^2 + bx + c)(x + d)$, then

$$cd = 1 \Rightarrow c = d = 1$$

and hence

$$c + bd = 0 \Rightarrow b = 1,$$

which contradicts $b + d = 1$. [Alternatively, we could note that $g(-d) = d^3 + d + 1 \neq 0$ for all $d \in Z_2$, so $g(x)$ cannot have a linear factor $(x + d)$.]

That is, if a and b are two polynomials in $\mathbb{Z}_2[x]/g$, their product can be zero $(\operatorname{mod} g)$ only if one of them is itself zero. Thus, $\mathbb{Z}_2[x]/g$ is a field with exactly 8 elements, corresponding to the 8 possible choices for the 3 polynomial coefficients in $\mathbb{Z}_2$.

Definition: The *order* of an element α of a finite field is the smallest natural number e such that $\alpha^e = 1$.

Definition: The *Euler indicator* or *Euler totient* function

$$\varphi(n) \doteq \{m \in \mathbb{N} : 1 \leq m \leq n, (m,n) = 1\}$$

is the number of positive integers less than or equal to n that are relatively prime (share no common factors).

- $\varphi(p) = p - 1$ for any prime number p.
- $\varphi(p^r) = p^r - p^{r-1}$ for any prime number p and any $r \in \mathbb{N}$ since p, $2p$, $3p$, ..., $p^{r-1}p$ all have a factor in common with p^r.

Remark: If we denote the set of integers in $\mathbb{Z}_n$ that are not zero divisors by $\mathbb{Z}_n^*$, we see for $n \geq 2$ that $\varphi(n) = |\mathbb{Z}_n^*|$.

- Here are the first 12 values of φ:

x	1	2	3	4	5	6	7	8	9	10	11	12
$\varphi(x)$	1	1	2	2	4	2	6	4	6	4	10	4

Remark: Note that

$$\varphi(1) + \varphi(2) + \varphi(3) + \varphi(6) = 1 + 1 + 2 + 2 = 6,$$

$$\varphi(1) + \varphi(2) + \varphi(3) + \varphi(4) + \varphi(6) + \varphi(12) = 1 + 1 + 2 + 2 + 2 + 4 = 12,$$

and $\varphi(1) + \varphi(p) = 1 + (p - 1) = p$ for any prime p.

Exercise: The *Chinese Remainder Theorem* implies that $\varphi(mn) = \varphi(m)\varphi(n)$ whenever $(m,n) = 1$. Use this result to prove for any $n \in \mathbb{N}$ that

$$\sum_{d|n} \varphi(d) = n.$$

Theorem A.4 (Primitive Element of a Field) *The nonzero elements of any finite field can all be written as a power of a single element.*

Proof: Given a finite field F of order q, let $1 \leq e \leq q-1$. Either there exists no elements in F of order e or there exists at least one element α of order e. In the latter case, α is a root of the polynomial $x^e - 1$ in $F[x]$; that is, $\alpha^e = 1$. Hence $(\alpha^n)^e = (\alpha^e)^n = 1$ for $n = 0, 1, 2, \ldots$. Since α has order e, we know that each of the roots α^n for $n = 1, 2, \ldots, e$ are distinct. Since $x^e - 1$ can have a most e zeros in $F[x]$, we then immediately know the factorization of the polynomial $x^e - 1$ in $F[x]$:

$$x^e - 1 = (x-1)(x-\alpha)(x-\alpha^2)\ldots(x-\alpha^{e-1}).$$

Thus, the only possible elements of order e in F are powers α^i for $1 \leq i < e$. However, if i and e share a common factor $n > 1$, then $(\alpha^i)^{e/n} = 1$ and the order of α^i would be less than or equal to e/n. So this leaves only the elements α^i where $(i, e) = 1$ as possible candidates for elements of order e. Note that the e powers of α are a subgroup (coset) of the multiplicative group G formed by the nonzero elements of F, so Lagrange's Theorem implies that e must divide the order of G, that is, $e|(q-1)$.

Consequently, the number of elements of order e, where e divides $q-1$ is either 0 or $\varphi(e)$. If the number of elements of order e were 0 for some divisor e of $q-1$, then the total number of nonzero elements in F would be less than $\sum_{d|(q-1)} \varphi(d) = q-1$, which is a contradiction. Hence, there exist elements in F of any order e that divides $q-1$, including $q-1$ itself. The distinct powers of an element of order $q-1$ are just the $q-1$ nonzero elements of F.

Definition: An element of order $q-1$ in a finite field F_q is called a *primitive element.*

Remark: Theorem A.4 states that the elements of a finite field F_q can be listed in terms of a primitive element, say α:

$$F_q = \{0, \alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{q-2}\}.$$

Remark: The fact that all elements in a field F_q can be expressed as powers of a primitive element can be exploited whenever we wish to multiply two elements together. We can compute the product $\alpha^i\alpha^j$ simply by determining which element can be expressed as α raised to the power $(i+j) \mod(q-1)$, in exactly the same manner as one uses a table of logarithms to perform multiplication.

Remark: The primitive element of a finite field F_q need not be unique. In fact, we see from the proof of Theorem A.4 that the number of such elements is $\varphi(q-1)$. Specifically, if α is a primitive element, then the powers α^i, for the $\varphi(p^r-1)$ values of i that are relatively prime to p^r-1, are also primitive elements.

Remark: A primitive element α of F_q satisfies the equation $\alpha^{q-1} = 1$, so that $\alpha^q = \alpha$, and has the highest possible order $(q-1)$. Note that $(\alpha^i)^{-1} = \alpha^{q-1-i}$.

Remark: If α is a primitive element of F_q, then α^{-1} is also a primitive element of F_q.

The fact that the primitive element α satisfies $\alpha^q = \alpha$ leads to the following corollary of Theorem A.4.

Corollary A.4.1 (Cyclic Nature of Fields) *Every element β of a finite field of order q is a root of the equation $\beta^q - \beta = 0$.*

Remark: In particular, Corollary A.4.1 states that every element β in a finite field F_{p^r} is a root of some polynomial $f(x) \in F_p[x]$.

Definition: Given an element β in a field F_{p^r}, the monic polynomial $m(x)$ in $F_p[x]$ of least degree with β as a root is called the *minimal polynomial* of β.

Theorem A.5 (Minimal Polynomial) *If $f(x) \in F_p[x]$ has β as a root, then $f(x)$ is divisible by the minimal polynomial of β.*

Proof: If $f(\beta) = 0$, then expressing $f(x) = q(x)m(x) + r(x)$ with $\deg r < \deg m$, we see that $r(\beta) = 0$. By the minimality of $\deg m$, we see that $r(x)$ is identically zero.

Corollary A.5.1 (Minimal Polynomials Divide $x^q - x$) *The minimal polynomial of an element of a field F_q divides $x^q - x$.*

Corollary A.5.2 (Irreducibility of Minimal Polynomial) *Let $m(x)$ be a monic polynomial in $F_p[x]$ that has β as a root. Then $m(x)$ is the minimal polynomial of $\beta \iff m(x)$ is irreducible in $F_p[x]$.*

Proof:

"$\Rightarrow$" If $m(x) = a(x)b(x)$, where a and b are of smaller degree, then $a(\beta)b(\beta) = 0$ implies that $a(\beta) = 0$ or $b(\beta) = 0$; this would contradict the minimality of $\deg m$. Thus $m(x)$ is irreducible.

"$\Leftarrow$" If $f(\beta) = 0$, then by Theorem A.5, $m(x)$ is divisible by the minimal polynomial of β. But since $m(x)$ is irreducible and monic, the minimal polynomial must be $m(x)$ itself.

Definition: A *primitive polynomial* of a field is the minimal polynomial of a primitive element of the field.

Q. How do we find the minimal polynomial of an element α^i in the field F_{p^r}?

A. The following theorems provide some assistance.

Theorem A.6 (Functions of Powers) *If $f(x) \in F_p[x]$, then $f(x^p) = [f(x)]^p$.*

Proof: Exercise.

Corollary A.6.1 (Root Powers) *If α is a root of a polynomial $f(x) \in F_p[x]$ then α^p is also a root of $f(x)$.*

Theorem A.7 (Reciprocal Polynomials) *In a finite field F_{p^r} the following statements hold:*

(a) If $\alpha \in F_{p^r}$ is a nonzero root of $f(x) \in F_p[x]$, then α^{-1} is a root of the reciprocal polynomial of $f(x)$.

(b) a polynomial is irreducible $\iff$ its reciprocal polynomial is irreducible.

(c) a polynomial is a minimal polynomial of a nonzero element $\alpha \in F_{p^r} \Rightarrow$ a (constant) multiple of its reciprocal polynomial is a minimal polynomial of α^{-1}.

(d) a polynomial is primitive $\Rightarrow$ a (constant) multiple of its reciprocal polynomial is primitive.

Proof: Exercise.

Suppose we want to find the minimal polynomial $m(x)$ of α^i in F_{p^r}. Identify the set of distinct elements $\{\alpha^i, \alpha^{ip}, \alpha^{ip^2}, \ldots\}$. The powers of α modulo $p^r - 1$ in this set form the *cyclotomic coset* of i. Suppose there are s distinct elements in this set. By Theorem A.6.1, each of these elements are roots of $m(x)$ and so the polynomial

$$f(x) = \prod_{k=0}^{s-1} (x - \alpha^{ip^k})$$

is a factor of $m(x)$. It can readily be shown that $f(x)$ has coefficients in F_p (that is, upon expanding all of the factors, all of the αs disappear!). Hence $f(x) \in F_p[x]$ and $f(\alpha^i) = 0$, so by Theorem A.5, we know also that $m(x)$ is a factor of $f(x)$. Thus, $m(x) = f(x)$.

Remark: Since the degree of the minimal polynomial $m(x)$ of α^i equals the number of elements s in the cyclotomic coset of α^i, we can sometimes use the previous theorems to help us quickly determine $m(x)$ without having to actually multiply out all of its factors. Note that, since $p^r = 1 \mod(p^r - 1)$, minimal polynomials in F_{p^r} have degree $s \le r$.

Remark: Every primitive polynomial of F_{p^r} has degree r and each of its roots is a primitive element of F_{p^r}.

- We now find the minimal polynomial of all 16 elements of the field $F_{2^4} = F_2[x]/(x^4 + x^3 + 1)$. The polynomial x is a primitive element of the field. Since x is a root of the irreducible polynomial $x^4 + x^3 + 1$, we know from Corollary A.5.2 that

x^4+x^3+1 is the minimal polynomial of x and hence is a primitive polynomial of F_{16}. The cyclotomic cosets consist of powers $i2^k \pmod{15}$ of each element α^i:

$$\{1,2,4,8\},$$

$$\{3,6,12,9\},$$

$$\{5,10\},$$

$$\{7,14,13,11\},$$

$$\{0\}.$$

The first cyclotomic coset corresponds to the primitive element $\alpha=x$, for which the minimal polynomial is x^4+x^3+1. This is also the minimal polynomial for the other powers of alpha in the cyclotomic coset containing 1, namely α^2, α^4, and α^8.

The reciprocal polynomial of x^4+x^3+1 is x^4+x+1; this is the minimal polynomial of the inverse elements $\alpha^{-i}=\alpha^{15-i}$ for $i=1,2,4,8$, that is, for α^{14}, α^{13}, α^{11}, and α^7. We see that these are just the elements corresponding to the second last coset.

We can also easily find the minimal polynomial of a^3, α^6, α^{12}, and α^9. Since $\alpha^{15}=1$, we observe that α^3 satisfies the equation $x^5-1=0$. We can factorize $x^5-1=(x-1)(x^4+x^3+x^2+x+1)$ and since $\alpha^3 \neq 1$, we know that α^3 must be a root of the remaining factor, $x^4+x^3+x^2+x+1$. Furthermore, since the cyclotomic coset corresponding to α^3 contains 4 elements, the minimal polynomial must have degree 4. So $x^4+x^3+x^2+x+1$ is in fact the minimal polynomial of α^3, α^6, α^9, and α^{12} (hence we have indirectly proven that $x^4+x^3+x^2+x+1$ is irreducible in $F_2[x]$).

Likewise, since the minimal polynomial of x^5 must be a factor of $x^3-1=(x-1)(x^2+x+1)$ with degree 2, we see that the minimal polynomial for these elements is x^2+x+1.

Finally, the minimal polynomial of the multiplicative unit $\alpha^0=1$ is just the first degree polynomial $x+1$. The minimal polynomial of 0 is x.

Remark: The cyclotomic cosets containing powers that are relatively prime to p^r-1 contain the $\varphi(p^r-1)$ primitive elements of F_{p^r}; their minimal polynomials are primitive and have degree r. Note that x^4+x^3+1 and x^4+x+1 are primitive polynomials of $F_2[x]/(x^4+x^3+1)$ and their roots comprise the $\varphi(15)=\varphi(5)\varphi(3)=4\cdot 2=8$ primitive elements of F_{p^r}. Even though the minimal polynomial of the element a^3 also has degree $r=4$, it is not a primitive polynomial, since $(a^3)^5=1$.

Remark: There is another interpretation of finite fields, as demonstrated by the following example. Consider the field $F_4 = F_2[x]/(x^2 + x + 1)$, which contains the elements $\{0, 1, x, x + 1\}$. Since the primitive element $\alpha = x$ satisfies the equation $x^2 + x + 1 = 0$, we could, using the quadratic formula, think of α as the complex number

$$\alpha = \frac{-1 + \sqrt{1-4}}{2} = -\frac{1}{2} + i\frac{\sqrt{3}}{2}.$$

The other root to the equation $x^2 + x + 1 = 0$ is the complex conjugate $\bar{\alpha}$ of α. That is, $x^2 + x + 1 = (x - \alpha)(x - \bar{\alpha})$. From this it follows that $1 = \alpha\bar{\alpha} = |\alpha|^2$ and hence $\alpha = e^{i\theta} = \cos\theta + i\sin\theta$ for some real number θ. In fact, we see that $\theta = 2\pi/3$. Thus $\alpha^3 = e^{3\theta i} = e^{2\pi i} = 1$. In this way, we have constructed a number α that is a primitive third root of unity, which is precisely what we mean when we say that α is a primitive element of F_4. The field F_4 may be thought of either as the set $\{0, 1, x, x + 1\}$ or as the set $\{0, 1, e^{2\pi i/3}, e^{-2\pi i/3}\}$. Similarly, the field $F_3 = \{0, 1, 2\}$ is isomorphic to $\{0, 1, -1\}$ and $F_5 = \{0, 1, 2, 3, 4\}$ is isomorphic to $\{0, 1, i, -1, -i\}$.

Bibliography

[Buchmann 2001] J. A. Buchmann, *Introduction to Cryptography*, Springer, New York, 2001.

[Hill 1997] R. Hill, *A First Course in Coding Theory*, Oxford University Press, Oxford, 1997.

[Koblitz 1994] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer, New York, 2nd edition, 1994.

[Mollin 2001] R. A. Mollin, *An Introduction to Cryptography*, Chapman & Hall/CRC, Boca Raton, 2001.

[Pless 1989] V. Pless, *Introduction to the Theory of Error-Correcting Codes*, Wiley, New York, 2nd edition, 1989.

[Rosen 2000] K. H. Rosen, *Elementary Number Theory and its applications*, Addison-Wesley, Reading, MA, 4th edition, 2000.

[van Lint 1991] J. van Lint, *Introduction to Coding Theory*, Springer, Berlin, 3rd edition, 1991.

[Welsh 2000] D. Welsh, *Codes and Cryptography*, Oxford University Press, Oxford, 2000.

Index