# Coding and Cryptography

T. W. Körner

July 23, 1998

*Transmitting messages across noisy channels is an important practical problem. Coding theory provides explicit ways of ensuring that messages remain legible even in the presence of errors. Cryptography on the other hand, makes sure that messages remain unreadable — except to the intended recipient. These complementary techniques turn out to have much in common mathematically.*

**Small print** The syllabus for the course is defined by the Faculty Board Schedules (which are minimal for lecturing and maximal for examining). I should **very much** appreciate being told of any corrections or possible improvements and might even part with a small reward to the first finder of particular errors. This document is written in LaTeX2e and stored in the file labelled `~twk/IIA/Codes.tex` on emu in (I hope) read permitted form. My e-mail address is `twk@dpmms`.

These notes are based on notes taken in the course of the previous lecturer Dr Pinch. Most of their virtues are his, most of their vices mine. Although the course makes use of one or two results from probability theory and a few more from algebra it is possible to follow the course successfully whilst taking these results on trust. There is a note on further reading at the end but [7] is a useful companion for the first three quarters of the course (up to the end of section 8) and [9] for the remainder. Please note that vectors are *row* vectors unless otherwise stated.

## Contents

## 1 What is an error correcting code?

Originally codes were a device for making messages hard to read. The study of such codes and their successors is called cryptography and will form the subject of the last quarter of these notes. However in the 19th century the optical[1] and then the electrical telegraph made it possible to send messages speedily but at a price. That price might be specified as so much per word or so much per letter. Obviously it made sense to have books of 'telegraph codes' in which one five letter combination QWADR, say, meant 'please book quiet room for two' and another QWNDR meant 'please book cheapest room for one'. Obviously, also, an error of one letter of a telegraph code could have unpleasant consequences.

Today messages are usually sent in as binary sequences like 01110010 . . . , but the transmission of each digit still costs money. Because of this, messages

[1]See *The Count of Monte Christo* and various Napoleonic sea stories. A statue to the inventor of the optical telegraph (semaphore) used to stand somewhere in Paris but seems to have disappeared.

are often 'compressed', that is shortened by removing redundant structure[2] In recognition of this fact we shall assume that we are asked to consider a collection of $m$ messages *each of which is equally likely.*

Our model is the following. When the 'source' produces one of the $m$ possible messages $\mu_i$ say, it is fed into a 'coder' which outputs a string $\mathbf{c}_i$ of $n$ binary digits. The string is then transmitted one digit at a time along a 'communication channel'. Each digit has probability $p$ of being mistransmitted (so that 0 becomes 1 or 1 becomes 0) independently of what happens to the other digits $[0 \leq p < 1/2]$. The transmitted message is then passed through a 'decoder' which either produces a message $\mu_j$ (where we hope that $j = i$) or an error message and passes it on to the 'receiver'.

**Exercise 1.1.** *Why do we not consider the case $1 \geq p > 1/2$? What if $p = 1/2$?*

An obvious example is the transmission of data from a distant space probe where (at least in the early days) the coder had to be simple and robust but the decoder could be as complex as the designer wished. On the other hand the decoder in a home CD player must be cheap but the encoding system which produces the disc can be very expensive.

For most of the time we shall concentrate our attention on a *code* $C \subseteq \{0,1\}^n$ consisting of the *codewords* $\mathbf{c}_i$. We say that $C$ has *size* $m = |C|$. If $m$ is large then we can carry a large number of possible messages (that

[2]In practice the situation is more complicated. Engineers distinguish between irreversible 'lossy compression' and reversible 'lossless compression'. For compact discs where bits are cheap the sound recorded can be reconstructed exactly. For digital sound broadcasting where bits are expensive the engineers make use of knowledge of the human auditory system (for example, the fact that we can not make out very soft noise in the presence of loud noises) to produce a result that might sound perfect (or nearly so) to us but which is in fact not. For mobile phones there can be greater loss of data because users do not demand anywhere close to perfection. For digital TV the situation is still more striking with reduction in data content from film to TV of anything up to a factor of 60. However medical and satellite pictures must be transmitted with no loss of data. Notice that lossless coding can be judged by absolute criteria but the merits of lossy coding can only be judged subjectively.

In theory lossless compression should lead to a signal indistinguishable (from a statistical point of view) from a random signal. In practice this is only possible in certain applications. As an indication of the kind of problem involved consider TV pictures. If we know that what is going to be transmitted is 'head and shoulders' or 'tennis matches' or 'cartoons' it is possible to obtain extraordinary compression ratios by 'tuning' the compression method to the expected pictures but then changes from what is expected can be disastrous. At present digital TV encoders merely expect the picture to consist of blocks which move at nearly constant velocity remaining more or less unchanged from frame to frame. In this as in other applications we know that after compression the signal still has non-trivial statistical properties but we do not know enough about them to exploit this.

is we can carry more information) but as $m$ increases it becomes harder to distinguish between different messages when errors occur. At one extreme, if $m = 1$, errors cause us no problems (since there is only one message) but no information is transmitted (since there is only one message). At the other extreme, if $m = 2^n$, we can transmit lots of messages but any error moves us from one codeword to another. We are led to the following rather natural definition.

**Definition 1.2.** *The information rate of $C$ is* $\dfrac{\log_2 m}{n}$.

Note that, since $m \leq 2^n$ the information rate is never greater than 1. Notice also that the values of the information rate when $m = 1$ and $m = 2^n$ agree with what we might expect.

How should our decoder work? We have assumed that all messages are equally likely and that errors are independent (this would not be true if, for example, errors occured in bursts[3]. Under these assumptions, a reasonable strategy for our decoder is to guess that the codeword sent is one which differs in the fewest places from the string of $n$ binary digits received. Here and elsewhere the discussion can be illuminated by the simple notion of a Hamming distance.

**Definition 1.3.** *If* $\mathbf{x},\ \mathbf{y} \in \{0,1\}^n$ *we write*

$$d(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{n} |x_j - y_j|$$

*and call* $d(\mathbf{x},\mathbf{y})$ *the Hamming distance between* $\mathbf{x}$ *and* $\mathbf{y}$.

**Lemma 1.4.** *The Hamming distance is a metric.*

We now do some very simple 1A probability.

[3]For the purposes of this course we note that this problem could be tackled by permuting the 'bits' of the message so that 'burst are spread out'. In theory we could do better than this by using the statistical properties of such bursts. In practice this may not be possible. In the paradigm case of mobile phones, the properties of the transmission channel are constantly changing and are not well understood. (Here the main restriction on interleaving is that it introduces time delays. One way round this is 'frequency hopping' in which several users constantly swap transmission channels 'dividing bursts among users.) One desirable property of codes for mobile phone users is that they should 'fail gracefully', that is that as the error rate for the channel rises the error rate for the receiver should not suddenly explode.

**Lemma 1.5.** *We work with coding and transmission scheme described above. Let* $\mathbf{c} \in C$ *and* $\mathbf{x} \in \{0,1\}^n$*.*

*(i) If* $d(\mathbf{c},\mathbf{x}) = r$ *then*

$$\Pr(\mathbf{x} \textit{ received given } \mathbf{c} \textit{ sent}) = p^r(1-p)^{n-r}.$$

*(ii) If* $d(\mathbf{c},\mathbf{x}) = r$ *then*

$$\Pr(\mathbf{c} \textit{ sent given } \mathbf{x} \textit{ received}) = A(\mathbf{x})p^r(1-p)^{n-r},$$

*where* $A(\mathbf{x})$ *does not depend on* $r$ *or* $\mathbf{c}$*.*

*(iii) If* $\mathbf{c}' \in C$ *and* $d(\mathbf{c}',\mathbf{x}) \geq d(\mathbf{c},\mathbf{x})$ *then*

$$\Pr(\mathbf{c} \textit{ sent given } \mathbf{x} \textit{ received}) \geq \Pr(\mathbf{c}' \textit{ sent given } \mathbf{x} \textit{ received})$$

*with equality if and only if* $d(\mathbf{c}',\mathbf{x}) = d(\mathbf{c},\mathbf{x})$*.*

The lemma just proved justifies our use, both explicit and implicit, throughout what follows of the so called *maximum likelihood* decoding rule.

**Definition 1.6.** *The maximum likelihood decoding rule states that a string* $\mathbf{x} \in \{0,1\}^n$ *received by a decoder should be decoded as (one of) the codewords at the smallest Hamming distance from* $\mathbf{x}$*.*

Notice that, although this decoding rule is mathematically attractive, it may be impractical if $C$ is large and there is no way of finding the codeword at the smallest distance from a particular $\mathbf{x}$ without making complete search through all the members of $C$.

## 2 Hamming's breakthrough

Although we have used simple probabilistic arguments to justify it, the maximum likelihood decoding rule will enable us to avoid probabilistic considerations for the rest of the course and to concentrate on algebraic and combinatorial considerations. The spirit of the course is exemplified in the next two definitions.

**Definition 2.1.** *We say that* $C$ *is* $d$ error detecting *if changing up to* $d$ *digits in a codeword never produces another codeword.*

**Definition 2.2.** *We say that* $C$ *is* $e$ error correcting *if knowing that a string of* $n$ *binary digits differs from some codeword of* $C$ *in at most* $e$ *places we can deduce the codeword.*

Here are two simple schemes.

*Repetition coding of length $n$.* We take codewords of the form

$$\mathbf{c} = (c, c, c, \dots, c)$$

with $c = 0$ or $c = 1$. The code $C$ is $n-1$ error detecting, and $\lfloor (n-1)/2 \rfloor$ error correcting. The maximum likelihood decoder chooses the symbol that occurs most often. (Here and elsewhere $\lfloor \alpha \rfloor$ is the largest integer $N \leq \alpha$ and $\lceil \alpha \rceil$ is the smallest integer $M \geq \alpha$.) Unfortunately the information rate is $1/n$ which is rather low[4].

*The paper tape code.* Here and elsewhere it is convenient to give $\{0, 1\}$ the structure of the field $\mathbb{F}_2 = \mathbb{Z}_2$ by using arithmetic modulo 2. The codewords have the form

$$\mathbf{c} = (c_1, c_2, c_3, \dots, c_n)$$

with $c_1$, $c_2$, ... , $c_{n-1}$ freely chosen elements of $\mathbb{F}_2$ and $c_n$ (the check digit) the element of $\mathbb{F}_2$ which gives

$$c_1 + c_2 + \cdots + c_{n-1} + c_n = 0.$$

The resulting code $C$ is 1 error detecting since, if $\mathbf{x} \in \mathbb{F}_2^n$ is obtained from $\mathbf{c} \in C$ by making a single error, we have

$$x_1 + x_2 + \cdots + x_{n-1} + x_n = 1.$$

However it is not error correcting since, if

$$x_1 + x_2 + \cdots + x_{n-1} + x_n = 1,$$

there are $n$ codewords $\mathbf{y}$ with Hamming distance $d(\mathbf{x}, \mathbf{y}) = 1$. The information rate is $(n-1)/n$. Traditional paper tape had 8 places per line each of which could have a punched hole or not so $n = 8$.

**Exercise 2.3.** *Machines tend to communicate in binary strings so this course concentrates on* binary alphabets *with two symbols. There is no particular difficulty in extending our ideas to* alphabets *with $n$ symbols though, of course, some tricks will only work for particular values of $n$. If you look at the inner title page of almost any recent book you will find its International Standard Book Number (ISBN). The ISBN uses single digits selected from 0, 1, ... , 8, 9 and X representing 10. Each ISBN consists of nine such digits $a_1$, $a_2$, ... , $a_9$ followed by a single check digit $a_{10}$ chosen so that*

$$10a_1 + 9a_2 + \cdots + 2a_9 + a_{10} \equiv 0 \mod 11. \qquad (*)$$

[4]Compare the chorus 'Oh no John, no John, no John no'.

*(In more sophisticated language our code $C$ consists of those elements $\mathbf{a} \in \mathbb{F}_{11}^{10}$ such that $\sum_{j=1}^{10}(11-j)a_j = 0$.)*

*(i) Find a couple of books*[5] *and check that $(*)$ holds for their ISBNs*[6].

*(ii) Show that $(*)$ will not work if you make a mistake in writing down one digit of an ISBN.*

*(iii) Show that $(*)$ may fail to detect two errors.*

*(iv) Show that $(*)$ will not work if you interchange two adjacent digits.*
*Errors of type (ii) and (iv) are the most common in typing*[7]. *In communication between publishers and booksellers both sides are anxious that errors should be detected but would prefer the other side to query errors rather than to guess what the error might have been.*

Hamming had access to an early electronic computer but was low down in the priority list of users. He would submit his programs encoded on paper tape to run over the weekend but often he would have his tape returned on Monday because the machine had detected an error in the tape. 'If the machine can detect an error' he asked himself 'why can the machine not correct it?' and he came up with the following scheme.
*Hamming's original code.* We work in $\mathbb{F}_2^7$. The codewords $\mathbf{c}$ are chosen to satisfy the three conditions.

$$\begin{aligned} c_1 + c_3 + c_5 + c_7 &= 0 \\ c_2 + c_3 + c_6 + c_7 &= 0 \\ c_4 + c_5 + c_6 + c_7 &= 0. \end{aligned}$$

By inspection we may choose $c_3$, $c_5$, $c_6$ and $c_7$ freely and then $c_1$, $c_2$ and $c_4$ are completely determined. The information rate is thus $4/7$.

Suppose that we receive the string $\mathbf{x} \in \mathbb{F}_2^7$. We form the *syndrome* $(z_1, z_2, z_4) \in \mathbb{F}_2^3$ given by

$$\begin{aligned} z_1 &= x_1 + x_3 + x_5 + x_7 \\ z_2 &= x_2 + x_3 + x_6 + x_7 \\ z_4 &= x_4 + x_5 + x_6 + x_7. \end{aligned}$$

If $\mathbf{x}$ is a codeword then $(z_1, z_2, z_4) = (0, 0, 0)$. If $\mathbf{c}$ is a codeword and the Hamming distance $d(\mathbf{x}, \mathbf{c}) = 1$ then the place in which $\mathbf{x}$ differs from $\mathbf{c}$ is given by $z_1 + 2z_2 + 4z_3$ (using ordinary addition, not addition modulo 2) as

[5] In case of difficulty your college library may be of assistance.

[6] In fact, $X$ is only used in the check digit place.

[7] Thus the 1997–8 syllabus for this course contains the rather charming misprint of 'snydrome' for 'syndrome'.

may be easily checked using linearity and a case by case study of the seven binary sequences $\mathbf{x}$ containing one 1 and six 0s. The Hamming code is thus 1 error correcting.

**Exercise 2.4.** *Suppose we use eight hole tape with the standard paper tape code and the probability that an error occurs at a particular place on the tape (i.e. a hole occurs where it should not or fails to occur where it should) is* $10^{-4}$. *A program requires about 10 000 lines of tape (each line containing eight places) using the paper tape code. Using the Poisson approximation, direct calculation (possible with a hand calculator but really no advance on the Poisson method) or otherwise show that the probability that the tape will be accepted as error free by the decoder is less than .04%.*

*Suppose now that we use the Hamming scheme (making no use of the last place in each line). Explain why the program requires about 17 500 lines of tape but that any particular line will be correctly decoded with probability about* $1 - (21 \times 10^{-8})$ *and the probability that the entire program will be correctly decoded is better than 99.6%.*

Hamming's scheme is easy to implement. It took a little time for his company to realise what he had done[8] but they were soon trying to patent it. In retrospect the idea of an error correcting code seems obvious (Hamming's scheme had actually been used as the basis of a Victorian party trick) and indeed two or three other people discovered it independently, but Hamming and his Co-discoverers had done more than find a clever answer to a question. They had asked an entirely new question and opened a new field for mathematics and engineering.

The times were propitious for the development of the new field. Before 1940 error correcting codes would have been luxuries, solutions looking for problems, after 1950 with the rise of the computer and new communication technologies they became necessities. Mathematicians and engineers returning from wartime duties in code breaking, code making and general communications problems were primed to grasp and extend the ideas. The mathematical engineer Claude Shannon may be considered the presiding genius of the new field.

## 3 General considerations

How good can error correcting and error detecting codes be? The following discussion is a natural development of the ideas we have already discussed.

[8]Experienced engineers came away from working demonstrations muttering 'I still don't believe it'.

**Definition 3.1.** *The minimum distance $d$ of a code is the smallest Hamming distance between distinct code words.*

We call a code of length $n$, size $m$ and distance $d$ a $[n, m, d]$ code. Less briefly, a set $C \subseteq \mathbb{F}_2^n$, with $|C| = m$ and

$$\min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C,\ \mathbf{x} \neq \mathbf{y}\} = d$$

is called a $[n, m, d]$ code. By an $[n, m]$ code we shall simply mean a code of length $n$ and size $m$.

**Lemma 3.2.** *A code of minimum distance $d$ can detect $d-1$ errors and correct $\lfloor \frac{d-1}{2} \rfloor$ errors. It cannot detect all sets of $d$ errors and cannot correct all sets of $\lfloor \frac{d-1}{2} \rfloor + 1$ errors.*

It is natural here and elsewhere to make use of the geometrical insight provided by the (closed) Hamming ball

$$B(\mathbf{x}, r) = \{\mathbf{y} : d(\mathbf{x}, \mathbf{y}) \leq r\}.$$

Observe that

$$|B(\mathbf{x}, r)| = |B(\mathbf{0}, r)|$$

for all $\mathbf{x}$ and so writing

$$V(n, r) = |B(\mathbf{0}, r)|$$

we know that $V(n, r)$ is the number of points in any Hamming ball of radius $r$. A simple counting argument shows that

$$V(n, r) = \sum_{j=0}^{r} \binom{n}{j}.$$

**Theorem 3.3 (Hamming's bound).** *If a code $C$ is $e$ error correcting then*

$$|C| \leq \frac{2^n}{V(n, e)}.$$

There is an obvious fascination (if not utility) in the search for codes which attain the exact Hamming bound.

**Definition 3.4.** *A code $C$ of length $n$ and size $m$ which can correct $e$ errors is called* perfect *if*

$$m = \frac{2^n}{V(n, e)}.$$

**Lemma 3.5.** *Hamming's original code is a* $[7, 16, 2]$ *code. It is perfect.*

It may be worth remarking in this context that if a code which can correct $e$ errors is perfect (i.e. has a perfect packing of Hamming balls of radius $e$ then the decoder must invariably give the wrong answer when presented with $e + 1$ errors. We note also that if (as will usually be the case) $2^n/V(n, e)$ is not an integer no perfect $e$ error correcting code can exist.

**Exercise 3.6.** *Even if* $2^n/V(n, e)$ *is an integer, no perfect code may exist.*
*(i) Verify that*

$$\frac{2^{90}}{V(90, 2)} = 2^{78}.$$

*(ii) Suppose that* $C$ *is a perfect 2 error correcting code of length* $90$ *and size* $2^{78}$*. Explain why we may suppose without loss of generality that* $\mathbf{0} \in C$*.*
*(iii) Let* $C$ *be as in (ii) with* $\mathbf{0} \in C$*. Consider the set*

$$X = \{\mathbf{x} \in \mathbb{F}_2^{90} : x_1 = 1,\ x_2 = 1,\ d(\mathbf{0}, \mathbf{x}) = 3\}.$$

*Show that corresponding to each* $\mathbf{x} \in X$ *we can find a unique* $\mathbf{c}(\mathbf{x}) \in C$ *such that* $d(\mathbf{c}(\mathbf{x}), \mathbf{x}) = 2$*.*
*(iv) Continuing with the argument of (iii) show that*

$$d(\mathbf{c}(\mathbf{x}), \mathbf{0}) = 5$$

*and that* $c_i(\mathbf{x}) = 1$ *whenever* $x_i = 1$*. By looking at* $d(\mathbf{c}(\mathbf{x}), \mathbf{c}(\mathbf{x}'))$ *for* $\mathbf{x}, \mathbf{x}' \in X$ *and invoking the Dirichlet pigeon-hole principle, or otherwise, obtain a contradiction.*
*(v) Conclude that there is no perfect* $[90, 2^{78}]$ *code.*

We obtained the Hamming bound which places an upper bound on how good a code can be by a *packing* argument. A *covering* argument gives us the GSV (Gilbert, Shannon, Varshamov) bound in the opposite direction. Let us write $A(n, d)$ for the size of the largest code with minimum distance $d$.

**Theorem 3.7 (Gilbert, Shannon, Varshamov).** *We have*

$$A(n, d) \geq \frac{2^n}{V(n, d - 1)}.$$

Until recently there were no general explicit constructions for codes which achieved the GVS bound (i.e. codes whose minimum distance $d$ satisfied the inequality $A(n,d)V(n,d-1) \geq 2^n$). Such a construction was finally found by Garcia and Stricheuth by using 'Goppa' codes.

Engineers are, of course, interested in 'best codes' of length $n$ for reasonably small values of $n$ but mathematicians are particularly interested in what happens as $n \to \infty$. To see what we should look at recall the so called weak law of large numbers (a simple consequence of Chebychev's inequality). In our case it yields the following result.

**Lemma 3.8.** *Consider the model of a noisy transmission channel used in this course in which each digit had probability $p$ of being wrongly transmitted independently of what happens to the other digits. If $\epsilon > 0$ then*

$$\Pr(\textit{number of errors in transmission for message of } n \textit{ digits } \geq (1+\epsilon)pn) \to 0$$

*as $n \to \infty$.*

By Lemma 3.2 a code of minimum distance $d$ can correct $\lfloor \frac{d-1}{2} \rfloor$ errors. Thus if we have an error rate $p$ and $\epsilon > 0$ we know that the probability that a code of length $n$ with error correcting capacity $\lceil (1+\epsilon)pn \rceil$ code will fail to correct a transmitted message falls to zero as $n \to \infty$. By definition the biggest code with minimum distance $\lceil 2(1+\epsilon)pn \rceil$ has size $A(n, \lceil 2(1+\epsilon)pn \rceil)$ and so has information rate $\log_2 A(n, \lceil 2(1+\epsilon)pn \rceil)/n$. Study of the behaviour of $\log_2 A(n, n\delta)/n$ will thus tell us how large an information rate is possible in the presence of a given error rate.

**Definition 3.9.** *If $0 < \delta < 1/2$ we write*

$$\alpha(\delta) = \limsup_{n\to\infty} \frac{\log_2 A(n, n\delta)}{n}.$$

**Definition 3.10.** *We define the entropy function $H : [0, 1/2) \to \mathbb{R}$ by $H(0) = 0$ and*

$$H(\delta) = -\delta \log_2(\delta) - (1-\delta)\log_2(1-\delta),$$

*for all $0 < \delta < 1/2$*

(Our function $H$ is a very special case of a general measure of disorder.)

**Theorem 3.11.** *With the definitions just given,*

$$1 - H(\delta) \leq \alpha(\delta) \leq 1 - H(\delta/2)$$

*for all $0 \leq \delta < 1/2$.*

Using the Hamming bound (Theorem 3.3) and the GSV bound (Theorem 3.7) we see that Theorem 3.11 follows at once from the following result.

**Theorem 3.12.** *We have*

$$\frac{\log_2 V(n, n\delta)}{n} \to H(\delta)$$

*as* $n \to \infty$.

Our proof of Theorem 3.12 depends, as one might expect on a version of Stirling's formula. We only need the very simplest version proved in 1A.

**Lemma 3.13 (Stirling).** *We have*

$$\log_e n! = n \log_e n - n + O(\log_2 n).$$

We combine this with the remark that

$$V(n, n\delta) = \sum_{0 \leq j \leq n\delta} \binom{n}{j},$$

and that very simple estimates give

$$\binom{n}{m} \leq \sum_{0 \leq j \leq n\delta} \binom{n}{j} \leq (m+1)\binom{n}{m}$$

where $m = \lfloor n\delta \rfloor$.

Although the GSV bound is very important a stronger result can be obtained for the error correcting power of the best long codes.

**Theorem 3.14 (Shannon's coding theorem).** *Suppose* $0 < p < 1/2$ *and* $\eta > 0$. *Then there exists an* $n_0(p, \eta)$ *such that for any* $n > n_0$ *we can find codes of length* $n$ *which have the property that (under our standard model) the probability that a codeword is mistaken is less than* $\epsilon$ *and have information rate* $1 - H(p) - \eta$.

[WARNING: Do not use this result until you have studied its proof. It is indeed a beautiful and powerful result but my statement conceals some traps for the unwary.]

Thus in our standard setup, by using sufficiently long code words, we can simultaneously obtain an information rate as close to $1 - H(p)$ as we please and and an error rate as close to 0 as we please. Shannon's proof uses the kind of ideas developed in this section with an extra pinch of probability (he chooses codewords *at random*) but I shall not give it in the course. There is a nice simple treatment in Chapter 3 of [9].

In view of Hamming's bound it is not surprising that it can also be shown that we cannot drive the error rate down to close to zero and maintain an information rate $1 - H_0 > 1 - H(p)$. To sum up, our standard set up, has *capacity* $1-H(p)$. We can *communicate reliably* at any fixed information rate below this capacity but not at any rate above. However, Shannon's theorem which tells us that rates less than $H(p)$ are possible is non-constructive and does not tell us how explicitly hoe to achieve these rates.

## 4 Linear codes

Just as $\mathbb{R}^n$ is vector space over $\mathbb{R}$ and $\mathbb{C}^n$ is vector space over $\mathbb{C}$ so $\mathbb{F}_2^n$ is vector space over the $\mathbb{F}_2$. (If you know about vector spaces over fields, so much the better, if not just follow the obvious paths.) A *linear code* is a subspace of $\mathbb{F}_2^n$. More formally we have the following definition.

**Definition 4.1.** *A linear code is a subset of* $\mathbb{F}_2^n$ *such that*
*(i)* $\mathbf{0} \in C$,
*(ii) if* $\mathbf{x}, \mathbf{y} \in C$ *then* $\mathbf{x} + \mathbf{y} \in C$.

Note that if $\lambda \in \mathbb{F}$ then $\lambda = 0$ or $\lambda = 1$ so that condition (i) of the definition just given guarantees that $\lambda\mathbf{x} \in C$ whenever $\mathbf{x} \in C$. We shall see that linear codes have many useful properties.

**Example 4.2.** *(i) The repetition code with*

$$\{C = \{\mathbf{x} : \mathbf{x} = (x, x, \dots x)\}$$

*is a linear code.*
*(ii) The paper tape code*

$$C = \left\{\mathbf{x} : \sum_{j=0}^{n} x_j = 0\right\}$$

*is a linear code.*
*(iii) Hamming's original code is a linear code.*

The verification is easy. In fact, examples (ii) and (iii) are 'parity check codes' and so automatically linear as we shall see from the next lemma.

**Definition 4.3.** *Consider a set $P$ in $\mathbb{F}_2^n$. We say that $C$ is the code defined by the set of* parity checks *$P$ if the elements of $C$ are precisely those $\mathbf{x} \in \mathbb{F}_2^n$ with*

$$\sum_{j=1}^{n} p_j x_j = 0$$

*for all $\mathbf{p} \in P$.*

**Lemma 4.4.** *If $C$ is code defined by parity checks then $C$ is linear.*

We now prove the converse result.

**Definition 4.5.** *If $C$ is a linear code we write $C^\perp$ for the set of $\mathbf{p} \in \mathbb{F}^n$ such that*

$$\sum_{j=1}^{n} p_j x_j = 0$$

*for all $\mathbf{x} \in C$.*

Thus $C^\perp$ is the set of parity checks satisfied by $C$.

**Lemma 4.6.** *If $C$ is a linear code then*
*(i) $C^\perp$ is a linear code,*
*(ii) $(C^\perp)^\perp \supseteq C$.*

We call $C^\perp$ the *dual code* to $C$.

In the language of the last part of the course on linear mathematics (P1), $C^\perp$ is the annihilator of $C$. The following is a standard theorem of that course.

**Lemma 4.7.** *If $C$ is a linear code in $\mathbb{F}_2^n$ then*

$$\dim C + \dim C^\perp = n.$$

Since the last part of P1 is not the most popular piece of mathematics in 1B we shall give an independent proof later (see the note after Lemma 4.13). Combining Lemma 4.6 (ii) with Lemma 4.7 we get the following corollaries.

**Lemma 4.8.** *If $C$ is a linear code then $(C^\perp)^\perp = C$.*

**Lemma 4.9.** *Every linear code is defined by parity checks.*

Our treatment of linear codes has been rather abstract. In order to put computational flesh on the dry theoretical bones we introduce the notion of a generator matrix.

**Definition 4.10.** *If $C$ is a linear code of length $n$ any $r \times n$ matrix whose rows form a basis for $C$ is called a* generator matrix *for $C$. We say that $C$ has* dimension *or* rank *$r$.*

**Example 4.11.** *As examples we can find generator matrices for the repetition code, the paper tape code and the original Hamming code.*

Remember that the Hamming code is the code of length 7 given by the parity conditions

$$\begin{aligned} x_1 + x_3 + x_5 + x_7 &= 0 \\ x_2 + x_3 + x_6 + x_7 &= 0 \\ x_4 + x_5 + x_6 + x_7 &= 0. \end{aligned}$$

By using row operations and column permutations we can use Gaussian elimination we can give a constructive proof of the following lemma.

**Lemma 4.12.** *Any linear code of length $n$ has (possibly after permuting the order of coordinates) a generator matrix of the form*

$$(I_r|B).$$

Notice that this means that any codeword $\mathbf{x}$ can be written as

$$(\mathbf{y}|\mathbf{z}) = (\mathbf{y}|\mathbf{y}B)$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_r)$ may be considered as the message and the vector $\mathbf{z} = \mathbf{y}B$ of length $n - r$ may be considered the check digits. Any code whose codewords can be split up in this manner is called *systematic*.

We now give a more computational treatment of parity checks.

**Lemma 4.13.** *If $C$ is a linear code of length $n$ with generator matrix $G$ then $\mathbf{a} \in C^\perp$ if and only if*

$$G\mathbf{a}^T = \mathbf{0}^T.$$

*Thus*

$$C^\perp = (\ker G)^T.$$

Thus using the rank, nullity theorem we get a second proof of Lemma 4.7.

Lemma 4.13 also enables us to characterise $C^\perp$.

**Lemma 4.14.** *If $C$ is a linear code of length $n$ and dimension $r$ with generator the $r \times n$ matrix $G$ then if $H$ is any $n \times n-r$ matrix with columns forming a basis of* $\ker G$ *we know that $H$ is a parity check matrix for $C$ and its transpose $H^T$ is a generator for $C^\perp$.*

**Example 4.15.** *(i) The dual of the paper tape code is the repetition code.*
*(ii) Hamming's original code has dual with generator*

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

We saw above that the codewords of a linear code can be written

$$(\mathbf{y}|\mathbf{z}) = (\mathbf{y}|\mathbf{y}B)$$

where $\mathbf{y}$ may be considered as the vector of message digits and $\mathbf{z} = \mathbf{y}B$ as the vector of check digits. Thus *encoders* for linear codes are easy to construct.

What about decoders? Recall that every linear code of length $n$ has a (non-unique) associated parity check matrix $H$ with the property that $\mathbf{x} \in C$ if and only if $\mathbf{x}H = \mathbf{0}$. If $z \in \mathbb{F}_2^n$ we define the *syndrome* of $\mathbf{z}$ to be $\mathbf{z}H$. The following lemma is mathematically trivial but forms the basis of the method of *syndrome decoding.*

**Lemma 4.16.** *Let $C$ be a linear code with parity check matrix $H$. If we are given $\mathbf{z} = \mathbf{x} + \mathbf{e}$ where $\mathbf{x}$ is a code word and the 'error vector' $\mathbf{e} \in \mathbb{F}_2^n$, then*

$$\mathbf{z}H = \mathbf{e}H.$$

Suppose we have tabulated the syndrome $\mathbf{u}H$ for all $\mathbf{u}$ with 'few' non-zero entries (say, all $\mathbf{u}$ with $d(\mathbf{u}, \mathbf{0}) \leq K$). If our decoder receives $\mathbf{z}$ it computes the syndrome $\mathbf{z}H$. If the syndrome is zero then $\mathbf{z} \in C$ and the decoder assumes the transmitted message was $\mathbf{z}$. If the syndrome of the received message is a non-zero vector $\mathbf{w}$ the decoder searches its list until it finds an $\mathbf{e}$ with $\mathbf{e}H = \mathbf{w}$. The decoder then assumes that the transmitted message was $\mathbf{x} = \mathbf{z} - \mathbf{e}$ (note that $\mathbf{z} - \mathbf{e}$ will always be a codeword, even if not the right one). This procedure will fail if $\mathbf{w}$ does not appear in the list but for this to be case at least $K + 1$ errors must have occured.

If we take $K = 1$, that is we only want a 1 error correcting code then writing $\mathbf{e}^{(i)}$ for the vector in $\mathbb{F}_2^n$ with 1 in the $i$th place and 0 elsewhere we see that the syndrome $\mathbf{e}^{(i)}H$ is the $i$th row of $H$. If the transmitted message $\mathbf{z}$ has syndrome $\mathbf{z}H$ equal to the $i$th row of $H$ then the decoder assumes that there has been an error in the $i$th place and nowhere else. (Recall the special case of Hamming's original code.)

If $K$ is large the task of searching the list of possible syndromes becomes onerous and, unless (as sometimes happens) we can find another trick, we find that 'decoding becomes dear' although 'encoding remains cheap'.

We conclude this section by looking at weights and the weight enumeration polynomial for a linear code. The idea here is to exploit the fact that if $C$ is linear code and $\mathbf{a} \in C$ then $\mathbf{a} + C = C$. Thus the 'view of $C$' from any codeword $\mathbf{a}$ is the same as the 'view of $C$' from the particular codeword $\mathbf{0}$.

**Definition 4.17.** *The* weight $w(\mathbf{x})$ *of a vector* $\mathbf{x} \in \mathbb{F}_2^n$ *is given by*

$$w(\mathbf{x}) = d(\mathbf{0}, \mathbf{x}).$$

**Lemma 4.18.** *If $w$ is the weight function on $\mathbb{F}_2^n$ and $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ then*
*(i)* $w(\mathbf{x}) \geq 0$,
*(ii)* $w(\mathbf{x}) = 0$ *if and only if* $\mathbf{x} = \mathbf{0}$,
*(iii)* $w(\mathbf{x}) + w(\mathbf{y}) \geq w(\mathbf{x} + \mathbf{y})$.

Since the minimum (non-zero) weight in a linear code is the same as the minimum (non-zero) distance we can talk about linear codes of minimum weight $d$ when we mean linear codes of minimum distance $d$.

The pattern of distances in a linear code is encapsulated in the weight enumeration polynomial.

**Definition 4.19.** *Let $C$ be a linear code of length $n$. We write $A_j$ for the number of codewords of weight $j$ and define the weight enumeration polynomial $W_C$ to be the polynomial in two real variables given by*

$$W_C(s,t) = \sum_{j=0}^{n} A_j s^j t^{n-j}.$$

Here are some simple properties of $W_C$.

**Lemma 4.20.** *Under the assumptions and with the notation of the Definition 4.19, the following results are true.*
*(i) $W_C$ is a homogeneous polynomial of degree $n$.*
*(ii) If $C$ has rank $r$ then $W_C(1,1) = 2^r$.*
*(iii) $W_C(0,1) = 1$.*
*(iv) $W_C(1,0)$ takes the value $0$ or $1$.*
*(v) $W_C(s,t) = W_C(t,s)$ for all $s$ and $t$ if and only if $W_C(1,0) = 1$.*

**Lemma 4.21.** *For our standard model of communication along an error prone channel with independent errors of probability $p$ and a linear code $C$ of length $n$,*

$$W_C(p, 1-p) = \Pr(\textit{receive a code word} \mid \textit{code word transmitted})$$

*and*

$$\Pr(\textit{receive incorrect code word} \mid \textit{code word transmitted}) = W_C(p, 1-p) - (1-p)^n.$$

**Example 4.22.** *(i) If $C$ is the repetition code, $W_C(s,t) = s^n + t^n$.*
*(ii) If $C$ is the paper tape code of length $n$, $W_C(s,t) = \frac{1}{2}((s+t)^n + (t-s)^n)$.*

Example 4.22 is a special case of the MacWilliams identity.

**Theorem 4.23 (MacWilliams identity).** *If $C$ is a linear code*

$$W_{C^\perp}(s,t) = 2^{-\dim C} W_C(t-s, t+s).$$

We shall not give a proof and even the result may be considered as starred.

## 5 Some general constructions

However interesting the theoretical study of codes may be to a pure mathematician, the engineer would prefer to have an arsenal of practical codes so that he or she can select the one most suitable for the job in hand. In this section we discuss the general Hamming codes and the Reed-Muller codes as well as some simple methods of obtaining new codes from old.

**Definition 5.1.** *Let $d$ be a strictly positive integer and let $n = 2^d - 1$. Consider the (column) vector space $D = \mathbb{F}_2^d$. Write down a $d \times n$ matrix $H$ whose columns are the $2^d - 1$ distinct non-zero vectors of $D$. The Hamming $(n, n-d)$ code is the linear code of length $n$ with $H$ as parity check matrix.*

Of course the Hamming $(n, n-d)$ code is only defined up to permutation of coordinates. We note that $H$ has rank $d$ so a simple use of the rank nullity theorem shows that our notation is consistent.

**Lemma 5.2.** *The Hamming $(n, n-d)$ code is a linear code of length $n$ and rank $d$ [$n = 2^d - 1$].*

**Example 5.3.** *The Hamming $(7, 4)$ code is the original Hamming code.*

The fact that any two rows of $H$ are linearly independent and a look at the appropriate syndromes gives us the main property of the general Hamming code.

**Lemma 5.4.** *The Hamming $(n, n-d)$ code has minimum weight* 3 *and is a perfect 1 error correcting code [$n = 2^d - 1$].*

Hamming codes are ideal in situations where very long strings of binary digits must be transmitted but the chance of an error in any individual digit is very small. (Look at Exercise 2.4.) It may be worth remarking that, apart from the Hamming codes there are only a few (and, in particular, a finite number) of examples of perfect codes known.

Here are a number of simple tricks for creating new codes from old.

**Definition 5.5.** *If $C$ is a code of length $n$ the* parity check extension *$C^+$ of $C$ is the code of length $n+1$ given by*

$$C^+ = \left\{ \mathbf{x} \in \mathbb{F}_2^{n+1} : (x_1, x_2, \dots, x_n) \in C,\ \sum_{j=1}^{n+1} x_j = 0 \right\}.$$

**Definition 5.6.** *If $C$ is a code of length $n$ the* truncation *$C^-$ of $C$ is the code of length $n-1$ given by*

$$C^- = \{((x_1, x_2, \dots, x_{n-1}) : (x_1, x_2, \dots, x_n) \in C \textit{ for some } x_n \in \mathbb{F}_2\}.$$

**Definition 5.7.** *If $C$ is a code of length $n$ the* shortening *(or* puncturing*) $C'$ of $C$ is the code of length $n-1$ given by*

$$C' = \{((x_1, x_2, \dots, x_{n-1}) : (x_1, x_2, \dots, x_{n-1}, 0) \in C\}.$$

**Lemma 5.8.** *If $C$ is linear so is its parity check extension $C^+$, its truncation $C^-$ and its shortening $C'$.*

How can we combine two linear codes $C_1$ and $C_2$? Our first thought might be to look at their direct sum

$$C_1 \oplus C_2 = \{(\mathbf{x}|\mathbf{y}) : \mathbf{x} \in C_1,\ \mathbf{y} \in C_2\},$$

but this is unlikely to be satisfactory.

**Lemma 5.9.** *If $C_1$ and $C_2$ are linear codes then we have the following relation between minimum distances.*

$$d(C_1 \oplus C_2) = \min(d(C_1), d(C_2)).$$

On the other hand if $C_1$ and $C_2$ satisfy rather particular conditions we can obtain a more promising construction.

**Definition 5.10.** *Suppose $C_1$ and $C_2$ are linear codes of length $n$ with $C_1 \supseteq C_2$ (i.e. with $C_2$ a subspace of $C_1$). We define the* bar product $C_1|C_2$ *of $C_1$ and $C_2$ to be the code of length $2n$ given by*

$$C_1|C_2 = \{(\mathbf{x}|\mathbf{x}+\mathbf{y}) : \mathbf{x} \in C_1,\ \mathbf{y} \in C_2\}.$$

**Lemma 5.11.** *Let $C_1$ and $C_2$ be linear codes of length $n$ with $C_1 \supseteq C_2$. Then the bar product $C_1|C_2$ is a linear code with*

$$\operatorname{rank} C_1|C_2 = \operatorname{rank} C_1 + \operatorname{rank} C_2.$$

*The minimum distance of $C_1|C_2$ satisfies the inequality*

$$d(C_1|C_2) \geq \min(2d(C_1), d(C_2)).$$

We now return to the construction of specific codes. Recall that the Hamming codes are suitable for situations when the error rate $p$ is very small and we want a high information rate. The Reed-Muller are suitable when the error rate is very high and we are prepared to sacrifice information rate. They were used by NASA for the radio transmissions from its planetary probes (a task which has been compared to signalling across the Atlantic with a child's torch[9]).

We start by considering the $2^d$ points $P_0,\ P_1,\ \dots,\ P_{2^d-1}$ of the space $X = \mathbb{F}_2^d$. Our code words will be of length $n = 2^d$ and will correspond to the indicator functions $\mathbb{I}_A$ on $X$. More specifically the possible code word $\mathbf{c}^A$ is given by

$$\begin{aligned} c_i^A &= 1 \qquad \text{if } P_i \in A \\ c_i^A &= 0 \qquad \text{otherwise.} \end{aligned}$$

for some $A \subseteq X$.

In addition to the usual vector space structure on $\mathbb{F}_2^n$ we define a new operation

$$\mathbf{c}^A \wedge \mathbf{c}^B = \mathbf{c}^{A \cap B}.$$

Thus if $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$,

$$(x_0, x_1, \dots, x_{n-1}) \wedge (y_0, y_1, \dots, y_{n-1}) = (x_0y_0, x_1y_1, \dots, x_{n-1}y_{n-1}).$$

[9]Strictly speaking the comparison is meaningless. However, it sounds impressive and that is the main thing.

Finally we consider the collection of $d$ hyperplanes

$$\pi_j = \{\mathbf{p} \in X : p_j = 0\ [1 \leq j \leq d]\}$$

in $\mathbb{F}_2^n$ and the corresponding indicator functions

$$\mathbf{h}^j = \mathbf{c}^{\pi_j},$$

together with the special vector

$$\mathbf{h}^0 = \mathbf{c}^X = (1, 1, \ldots, 1).$$

**Exercise 5.12.** *Suppose that* $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_2^n$ *and* $A, B \subseteq X$.

*(i) Show that* $\mathbf{x} \wedge \mathbf{y} = \mathbf{y} \wedge \mathbf{x}$.

*(ii) Show that* $(\mathbf{x} + \mathbf{y}) \wedge \mathbf{z} = \mathbf{x} \wedge \mathbf{z} + \mathbf{y} \wedge \mathbf{z}$.

*(iii) Show that* $\mathbf{h}^0 \wedge \mathbf{x} = \mathbf{x}$.

*(iv) If* $\mathbf{c}^A + \mathbf{c}^B = \mathbf{c}^E$ *find* $E$ *in terms of* $A$ *and* $B$.

*(v) If* $\mathbf{h}^0 + \mathbf{c}^A = \mathbf{c}^E$ *find* $E$ *in terms of* $A$.

We refer to $\mathcal{A}_0 = \{\mathbf{h}^0\}$ as the set of terms of order zero. If $\mathcal{A}_k$ is the set of terms of order at most $k$ then the set $\mathcal{A}_{k+1}$ of terms of order at most $k+1$ is defined by

$$\mathcal{A}_{k+1} = \{\mathbf{a} \wedge \mathbf{h}^j : \mathbf{a} \in \mathcal{A}_k,\ 1 \leq j \leq n\}.$$

Less formally but more clearly the elements of order 1 are the $\mathbf{h}^i$, the elements of order 2 are the $\mathbf{h}^i \wedge \mathbf{h}^j$ with $i < j$, the elements of order 3 are the $\mathbf{h}^i \wedge \mathbf{h}^j \wedge \mathbf{h}^k$ with $i < j < k$ and so on.

**Definition 5.13.** *Using the notation established above, the Reed-Miller code* $RM(d, r)$ *is the linear code (i.e. subspace of* $\mathbb{F}^n$*) generated by the terms of order* $r$ *or less.*

Although the formal definition of the Reed-Miller codes looks pretty impenetrable at first sight, once we have looked at sufficiently many examples it should become clear what is going on.

**Example 5.14.** *(i) The* $RM(3, 0)$ *code is the repetition code of length* 8.

*(ii) The* $RM(3, 1)$ *code is the parity check extension of Hamming's original code.*

*(iii) The* $RM(3, 2)$ *code is the paper tape code of length* 8.

*(iii) The* $RM(3, 3)$ *code is the trivial code consisting of all the elements of* $\mathbb{F}_2^3$.

We now prove the key properties of the Reed-Miller codes. We use the notation established above.

**Theorem 5.15.** *(i) The elements of order $d$ or less (that is the collection of all possible wedge products formed from the $\mathbf{h}^i$) span $\mathbb{F}_2^n$.*

*(ii) The elements of order $d$ or less are linearly independent.*

*(iii) The dimension of the Reed-Miller code $RM(d,r)$ is*

$$\binom{d}{0}+\binom{d}{1}+\binom{d}{2}+\dots+\binom{d}{r}.$$

*(iv) Using the bar product notation we have*

$$RM(d,r) = RM(d-1,r)|RM(d-1,r-1).$$

*(v) The minimum weight of $RM(d,r)$ is exactly $2^{d-r}$.*

**Exercise 5.16.** *The Mariner mission to Mars used the $RM(5,1)$ code. What was its information rate. What proportion of errors could it correct in a single code word?*

**Exercise 5.17.** *Show that the $RM(d,d-2)$ code is the parity extension code of the Hamming $(N,N-d)$ code with $N=2^d-1$.*

## 6 Polynomials and fields

This section is starred and will not be covered in lectures. Its object is to make plausible the few facts from modern[10] algebra that we shall need. They were covered, along with much else, in the course O4 (Groups, rings and fields) but attendance at that course is no more required for this course than is reading Joyce's *Ulysses* before going for a night out at an Irish pub. Anyone capable of criticising the imprecision and general slackness of the account that follows obviously can do better themselves and should omit this section.

A field $K$ is an object equipped with addition and multiplication which follow the same rules as do addition and multiplication in $\mathbb{R}$. The only rule which will cause us trouble is

$$\text{If } x \in K \text{ and } x \neq 0 \text{ then we can find } y \in K \text{ such that } xy = 1. \qquad \bigstar$$

Obvious examples of fields include $\mathbb{R}$, $\mathbb{C}$ and $\mathbb{F}_2$.

We are particularly interested in polynomials over fields but here an interesting difficulty arises.

[10]Modern, that is, in 1850.

**Example 6.1.** *We have $t^2 + t = 0$ for all $t \in \mathbb{F}_2$.*

To get round this, we distinguish between the polynomial in the 'indeterminate' $X$

$$P(X) = \sum_{j=0}^{n} a_j X^j$$

with coefficients in $a_j \in K$ and its evaluation $P(t) = \sum_{j=0}^{n} a_j t^j$ for some $t \in K$. We manipulate polynomials in $X$ according to the standard rules for polynomials but say that

$$\sum_{j=0}^{n} a_j X^j = 0$$

if and only if $a_j = 0$ for all $j$. Thus $X^2 + X$ is a non-zero polynomial over $\mathbb{F}_2$ all of whose values are zero.

The following result is familiar, in essence, from school mathematics.

**Lemma 6.2 (Remainder theorem).** *(i) If $P$ is a polynomial over a field $K$ and $a \in K$ then we can find a polynomial $Q$ and an $r \in K$ such that*

$$P(X) = (X - a)Q(X) + r.$$

*(ii) If $P$ is a polynomial over a field $K$ and $a \in K$ is such that $P(a) = 0$ then we can find a polynomial $Q$ such that*

$$P(X) = (X - a)Q(X).$$

The key to much of the elementary theory of polynomials lies in the fact that we can apply Euclid's algorithm to obtain results like the following.

**Theorem 6.3.** *Suppose that $\mathcal{P}$ is a set of polynomials,which contains at least one non-zero polynomial and has the following properties.*

*(i) If $Q$ is any polynomial and $P \in \mathcal{P}$ then the product $PQ \in \mathcal{P}$.*

*(ii) If $P_1, P_2 \in \mathcal{P}$ then $P_1 + P_2 \in \mathcal{P}$.*

*Then we can find a non-zero $P_0 \in \mathcal{P}$ which divides every $P \in \mathcal{P}$.*

*Proof.* Consider a non zero polynomial $P_0$ of smallest degree in $\mathcal{P}$. □

Recall that the polynomial $P(X) = X^2 + 1$ has no roots in $\mathbb{R}$ (that is $P(t) \neq 0$ for all $t \in \mathbb{R}$). However by considering the collection of formal expressions $a + bi$ $[a, b \in \mathbb{R}]$ with the obvious formal definitions of addition and multiplication and subject to the further condition $i^2 + 1 = 0$ we obtain a field $\mathbb{C} \supseteq \mathbb{R}$ in which $P$ has a root (since $P(i) = 0$). We can perform a similar trick with other fields.

**Example 6.4.** *If $P(X) = X^2 + X + 1$ then $P$ has no roots in $\mathbb{F}_2$. However if we consider*

$$\mathbb{F}_2[\omega] = \{0,\ 1,\ \omega,\ 1 + \omega\}$$

*with obvious formal definitions of addition and multiplication and subject to the further condition $\omega^2 + \omega + 1 = 0$ then $\mathbb{F}_2[\omega]$ is a field containing $\mathbb{F}_2$ in which $P$ has root (since $P(\omega) = 0$).*

*Proof.* The only thing we really need prove is that $\mathbb{F}_2[\omega]$ is a field and to do that the only thing we need to prove is that $\bigstar$ holds. Since

$$(1 + \omega)\omega = 1$$

this is easy. $\square$

In order to state a correct generalisation of the ideas of the previous paragraph we need a preliminary definition.

**Definition 6.5.** *If $P$ is a polynomial over a field $K$ we say that $P$ is* reducible *if there exists a non-constant polynomial $Q$ of degree strictly less than $P$ which divides $P$. If $P$ is a non-constant polynomial which is not reducible then $P$ is* irreducible.

**Theorem 6.6.** *If $P$ is an irreducible polynomial of degree $n \geq 2$ over a field $K$. then $P$ has no roots in $K$. However if we consider*

$$K[\omega] = \left\{ \sum_{j=0}^{n-1} a_j \omega^j : a_j \in K \right\}$$

*with the obvious formal definitions of addition and multiplication and subject to the further condition $P(\omega) = 0$ then $K[\omega]$ is a field containing $K$ in which $P$ has root.*

*Proof.* The only thing we really need prove is that $K[\omega]$ is a field and to do that the only thing we need to prove is that $\bigstar$ holds. Let $Q$ be a non-zero polynomial of degree at most $n - 1$. Since $P$ is irreducible, the polynomials $P$ and $Q$ have no common factor of degree 1 or more. Hence, by Euclid's algorithm we can find polynomials $R$ and $S$ such that

$$R(X)Q(X) + S(X)P(X) = 1$$

and so $R(\omega)Q(\omega) + S(\omega)P(\omega) = 1$. But $P(\omega) = 0$ so $R(\omega)Q(\omega) = 1$ and we have proved $\bigstar$. $\square$

In a proper algebra course we would simply define

$$K[\omega] = K[X]/(P(X))$$

where $(P(X))$ is the ideal generated by $P(X)$. This is a cleaner procedure which avoids the use of such phrases as 'the obvious formal definitions of addition and multiplication' but the underlying idea remains the same.

**Lemma 6.7.** *If $P$ is polynomial over a field $K$ which does not factorise completely into linear factors then we can find a field $L \supseteq K$ in which $P$ has more linear factors.*

*Proof.* Factor $P$ into irreducible factors and choose a factor $Q$ which is not linear. By Theorem 6.6 we can find a field $L \supseteq K$ in which $Q$ has a root $\alpha$ say and so by Lemma 6.2 a linear factor $X - \alpha$. Since any linear factor of $P$ in $K$ remains a factor in the bigger field $L$ we are done. $\square$

**Theorem 6.8.** *If $P$ is polynomial over a field $K$ then we can find a field $L \supseteq K$ in which $P$ factorises completely into linear factors.*

We shall be interested in finite fields (that is fields $K$ with only a finite number of elements). A glance at our method of proving Theorem 6.8 shows that the following result holds.

**Lemma 6.9.** *If $P$ is polynomial over a finite field $K$ then we can find a finite field $L \supseteq K$ in which $P$ factorises completely.*

In this context, we note yet another useful simple consequence of Euclid's algorithm.

**Lemma 6.10.** *Suppose that $P$ is an irreducible polynomial over a field $K$ which has a linear factor $X - \alpha$ in some field $L \supseteq K$. If $Q$ is a polynomial over $K$ which has the factor $X - \alpha$ in $L$ then $P$ divides $Q$.*

We shall need a lemma on repeated roots.

**Lemma 6.11.** *Let $K$ be field. If $P(X) = \sum_{j=0}^{n} a_j X^j$ is a polynomial over $K$ we define $P'(X) = \sum_{j=1}^{n-1} a_j X^j$.*

*(i) If $P$ and $Q$ are polynomials $(P+Q)' = P'+Q'$ and $(PQ)' = P'Q+PQ'$.*

*(ii) If $P$ and $Q$ are polynomials with $P(X) = (X-a)^2 Q(X)$ then*

$$P'(X) = 2(X-a)Q(X) + (X-a)^2 Q'(X).$$

*(iii) If $P$ is divisible by $(X-a)^2$ then $P(a) = P'(a) = 0$.*

If $L$ is a field containing $\mathbb{F}_2$ then $2y = (1+1)y = 0y = 0$ for all $y \in L$. We can thus deduce the following result which will be used in the next section.

**Lemma 6.12.** *If $L$ is a field containing $\mathbb{F}_2$ and $n$ is an odd integer then $X^n - 1$ can have no repeated linear factors as a polynomial over $L$.*

We also need a result on roots of unity given as part (v) of the next lemma.

**Lemma 6.13.** *(i) If $G$ is a finite Abelian group and $x, y \in G$ have coprime orders $r$ and $s$ then $xy$ has order $rs$.*

*(ii) If $G$ is a finite Abelian group and $x, y \in G$ have orders $r$ and $s$ then we can find an element $z$ of $G$ with order the lowest common multiple of $r$ and $s$.*

*(iii) If $G$ is a finite Abelian group then there exists an $N$ and an $h \in G$ such that $h$ has order $N$ and $g^N = e$ for all $g \in G$.*

*(iv) If $G$ is a finite subset of a field $K$ which is a group under multiplication then $G$ is cyclic.*

*(v) Suppose $n$ is an odd integer. If $L$ is a field containing $\mathbb{F}_2$ such that $X^n - 1$ factorises completely into linear terms then we can find an $\omega \in L$ such that the roots of $X^n - 1$ are $1$, $\alpha$, $\alpha^2$, $\dots$ $\alpha^{n-1}$. (We call $\alpha$ a* primitive *$n$th root of unity.)*

*Proof.* (ii) Consider $z = x^u y^v$ where $u$ is a divisor of $r$, $v$ is a divisor of $s$, $r/u$ and $s/v$ are coprime and $rs/(uv) = \operatorname{lcm}(r, s)$.

(iii) Let $h$ be an element of highest order in $G$ and use (ii).

(iv) By (iii) we can find an integer $N$ and a $h \in G$ such that $h$ has order $N$ and any element $g \in G$ satisfies $g^N = 1$. Thus $X^N - 1$ has a linear factor $X - g$ for each $g \in G$ and so $\prod_{g \in G}(X - g)$ divides $X^N - 1$. It follows that the order $|G|$ of $G$ cannot exceed $N$. But by Lagrange's theorem $N$ divides $G$. Thus $|G| = N$ and $g$ generates $G$.

(v) Observe that $G = \{\omega : \omega^n = 1\}$ is an Abelian group with exactly $n$ elements (since $X^n - 1$ has no repeated roots) and use (iv). □

Here is another interesting consequence of Lemma 6.13 (iv)

**Lemma 6.14.** *If $K$ is a field with $2^n$ elements containing $\mathbb{F}_2$ then there is an element $k$ of $K$ such that*

$$K = \{0\} \cup \{k^r : 0 \leq r \leq 2^n - 2$$

*and $k^{2^n - 1} = 1$.*

*Proof.* Observe that $K \setminus \{0\}$ forms a group under multiplication. □

With this hint it is not hard to show that there is indeed a field with $2^n$ elements containing $\mathbb{F}_2$.

**Lemma 6.15.** *Let $L$ be some field containing $\mathbb{F}_2$ in which $X^{2^n-1} - 1 = 0$ factorises completely. Then*

$$K = \{x \in L : x^{2^n} = x\}$$

*is a field with $2^n$ elements containing $\mathbb{F}_2$.*

Lemma 6.14 shows that that there is (up to field isomorphism) only one field with $2^n$ elements containing $\mathbb{F}_2$. We call it $\mathbb{F}_{2^n}$. We call an element $k$ with the properties given in Lemma 6.14 a *primitive element* of $\mathbb{F}_{2^n}$.

## 7 Cyclic codes

In this section we discuss a subclass of linear codes, the so called *cyclic codes.*

**Definition 7.1.** *A linear code $C$ in $\mathbb{F}_2^n$ is called cyclic if*

$$(a_0, a_1, \dots, a_{n-2}, a_{n-1}) \in C \Rightarrow (a_1, a_2, \dots, a_{n-1}, a_0) \in C.$$

Let us establish a correspondence between $\mathbb{F}_2^n$ and the polynomials on $\mathbb{F}_2$ modulo $X^n - 1$ by setting

$$P_{\mathbf{a}} = \sum_{j=0}^{n} a_j X^j$$

whenever $\mathbf{a} \in \mathbb{F}_2$. (Of course, $X^n - 1 = X^n + 1$ but in this context the first expression seems more natural.)

**Exercise 7.2.** *With the notation just established show that*
*(i) $P_{\mathbf{a}} + P_{\mathbf{b}} = P_{\mathbf{a}+\mathbf{b}}$,*
*(ii) $P_{\mathbf{a}} = 0$ if and only if $\mathbf{a} = \mathbf{0}$.*

**Lemma 7.3.** *A code $C$ in $\mathbb{F}_2^n$ is cyclic if and only if $\mathcal{P}_C = \{P_{\mathbf{a}} : \mathbf{a} \in C\}$ satisfies the following two conditions (working modulo $X^n - 1$).*
*(i) If $f, g \in \mathcal{P}_C$ then $f + g \in \mathcal{P}_C$.*
*(ii) If $f \in \mathcal{P}_C$ and $g$ is any polynomial then the product $fg \in \mathcal{P}_C$.*

(In the language of abstract algebra, $C$ is cyclic if and only if $\mathcal{P}_C$ is an *ideal* of the quotient ring $\mathbb{F}_2[X]/(X^n - 1)$.)

From now on we shall talk of the code word $f(X)$ when we mean the code word $\mathbf{a}$ with $P_{\mathbf{a}}(X) = f(X)$. An application of Euclid's algorithm gives the following useful result.

**Lemma 7.4.** *A code $C$ of length $n$ is cyclic if and only if (working modulo $X^n - 1$, and using the conventions established above) there exists a polynomial $g$ such that*

$$C = \{f(X)g(X) : f \text{ a polynomial}\}$$

(In the language of abstract algebra, $\mathbb{F}_2[X]$ is a Euclidean domain and so a principal ideal domain. Thus the quotient $\mathbb{F}_2[X]/(X^n - 1)$ is a principal ideal domain.) We call $g(X)$ a generator polynomial for $C$

**Lemma 7.5.** *A polynomial $g$ is a generator for a cyclic code of length $n$ if and only if it divides $X^n - 1$.*

Thus we must seek generators among the factors of $X^n - 1 = X^n + 1$. If there are no conditions on $n$ the result can be rather disappointing.

**Exercise 7.6.** *If we work with polynomials over $\mathbb{F}_2$ then*

$$X^{2^r} + 1 = (X + 1)^{2^r}.$$

In order to avoid this problem and to be able to make use of Lemma 6.12 we shall take $n$ odd from now on. (In this case the cyclic codes are said to be separable.) Notice that the task of finding irreducible factors (that is factors with no further factorisation) is a finite one.

**Lemma 7.7.** *Consider codes of length $n$ Suppose that $g(X)h(X) = X^n - 1$. Then $g$ is a generator of a cyclic code $C$ and $h$ is a generator for a cyclic code which is the reverse of $C^\perp$.*

As an immediate corollary we have the following remark.

**Lemma 7.8.** *The dual of a cyclic code is itself cyclic.*

**Lemma 7.9.** *If a cyclic code $C$ of length $n$ has generator $g$ of degree $n - r$ then $g(X)$, $Xg(X)$, … , $X^{r-1}g(X)$ form a basis for $C$.*

Cyclic codes are thus easy to specify (we just need to write down the generator polynomial $g$) and to encode.

**Example 7.10.** *There are three cyclic codes of length 7 corresponding to irreducible polynomials of which two are versions of Hamming's original code.*

We know that $X^n + 1$ factorises completely over some larger finite field and, since $n$ is odd, we know by Lemma 6.12 that it has no repeated factors. The same is therefore true for any polynomial dividing it.

**Lemma 7.11.** *Suppose that $g$ is a generator of a cyclic code $C$ of odd length $n$. Suppose further that $g$ factorises completely into linear factors in some field $K$ containing $\mathbb{F}_2$. If $g = g_1 g_2 \dots g_k$ with each $g_j$ irreducible over $\mathbb{F}_2$ and $A$ is a set consisting only of the roots of the $g_j$ and containing at least one root of each $g_j$ $[1 \le j \le k]$, then*

$$C = \{f \in \mathbb{F}_2[X] : f(\alpha) = 0 \text{ for all } \alpha \in A\}.$$

**Definition 7.12.** *$A$* defining set *for a cyclic code $C$ is a set $A$ of elements in some field $K$ containing $\mathbb{F}_2$ such that $f \in \mathbb{F}_2[X]$ belongs to $C$ if and only if $f(\alpha) = 0$ for all $\alpha \in A$.*

(Note that, if $C$ has length $n$, $A$ must be a set of zeros of $X^n - 1$.)

**Lemma 7.13.** *Suppose that*

$$A = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$$

*is a defining set for a cyclic code $C$ in some field $K$ containing $\mathbb{F}_2$. Let $B$ be the $r \times n$ matrix over $K$ whose $j$th column is*

$$(1, \alpha_j, \alpha_j^2, \dots, \alpha_j^{n-1})^T$$

*Then a vector $\mathbf{a} \in \mathbb{F}_2^n$ is a code word in $C$ if and only if*

$$\mathbf{a}B = \mathbf{0}$$

*in $K$.*

The columns in $B$ are not parity checks in the usual sense since the code entries lie in $\mathbb{F}_2$ and the computations take place in the larger field $K$.

With this background we can discuss a famous family of codes known as the BCH (Bose, Ray-Chaudhuri, Hocquenghem) codes. Recall that a primitive $n$th root of unity is an root $\alpha$ of $X^n - 1 = 0$ such that every root is a power of $\alpha$

**Definition 7.14.** *Suppose that $n$ is odd and $K$ is a field containing $\mathbb{F}_2$ in which $X^n - 1$ factorises into linear factors. Suppose that $\alpha \in K$ is a primitive $n$th root of unity. A cyclic code $C$ with defining set*

$$A = \{\alpha, \alpha^2, \dots, \alpha^{\delta-1}\}$$

*is a* BCH code of design distance *$\delta$.*

Note that the rank of $C$ will be $n-k$ where $k$ is the degree of the product those irreducible factors of $X^n-1$ over $\mathbb{F}$ which have a zero in $A$. Notice also that $k$ may be very much larger than $\delta$.

**Example 7.15.** *(i) If $K$ is a field containing $\mathbb{F}_2$ then $(a+b)^2=a^2+b^2$ for all $a,b\in K$.*

*(ii) If $P\in\mathbb{F}_2[X]$ and $K$ is a field containing $\mathbb{F}_2$ then $P(a)^2=P(a^2)$ for all $a\in K$.*

*(iii) Let $K$ be a field containing $\mathbb{F}_2$ in which $X^7-1$ factorises into linear factors. If $\beta$ is a root of $X^3+X+1$ in $K$ then $\beta$ is a primitive root of unity and $\beta^2$ is also a root of $X^3+X+1$.*

*(iv) We continue with the notation (iii). The BCH code with $\{\beta,\beta^2\}$ as defining set is Hamming's original (7,4) code.*

The next theorem contains the key fact about BCH codes.

**Theorem 7.16.** *The minimum distance for a BCH code is at least as great as the design distance.*

Our proof of Theorem 7.16 relies on showing that the matrix $B$ of Lemma 7.13 is non-singular for a BCH. To do this we use a result which every undergraduate knew in 1950.

**Lemma 7.17 (The van der Monde determinant).** *We work over a field $K$. The determinant*

$$\begin{vmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_n \\ x_1^2 & x_2^2 & x_3^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & \dots & x_n^{n-1} \end{vmatrix} = \prod_{1\le j<i\le n}(x_i-x_j)$$

How can we construct a decoder for a BCH code? From now on until the end of this section we shall suppose that we are using the BCH code $C$ described in Definition 7.14. In particular $C$ will have length $n$ and defining set

$$A=\{\alpha,\alpha^2,\dots,\alpha^{\delta-1}\}$$

where $\alpha$ is a primitive $n$th root of unity in $K$. Let $t$ be the largest integer with $2t+1\le\delta$. We show how we can correct up to $t$ errors.

Suppose that a codeword $\mathbf{c}=(c_0,c_1,\dots,c_{n-1})$ is transmitted and that the string received is $\mathbf{r}$. We write $\mathbf{e}=\mathbf{r}-\mathbf{c}$ and assume that

$$\mathcal{E}=\{0\le j\le n-1: e_j\ne 0\}$$

has no more than $t$ members. In other words $\mathbf{e}$ is the error vector and we assume that there are no more than $t$ errors. We write

$$c(X) = \sum_{j=0}^{n-1} c_j X^j,$$

$$r(X) = \sum_{j=0}^{n-1} r_j X^j,$$

$$e(X) = \sum_{j=0}^{n-1} e_j X^j.$$

**Definition 7.18.** *The* error locator polynomial *is*

$$\sigma(X) = \prod_{j \in \mathcal{E}} (1 - \alpha^j X)$$

*and the* error co-locator *is*

$$\omega(X) = \sum_{i=0}^{n-1} e_i \alpha^i \prod_{j \in \mathcal{E},\ j \neq i} (1 - \alpha^j X).$$

Informally we write

$$\omega(X) = \sum_{i=0}^{n-1} e_i \alpha^i \frac{\sigma(X)}{1 - \alpha^i X}.$$

We take $\omega(X) = \sum_j \omega_j X^j$ and $\sigma(X) = \sum_j \sigma_j X^j$. Note that $\omega$ has degree at most $t-1$ and $\sigma$ degree at most $t$. Note that we know that $\sigma_0 = 1$ so both the polynomials $\omega$ and $\sigma$ have $t$ unknown coefficients.

**Lemma 7.19.** *If the error location polynomial is given the value of* $\mathbf{e}$ *and so of* $\mathbf{c}$ *can be obtained directly.*

We wish to make use of relations of the form

$$\frac{1}{1 - \alpha^j X} = \sum_{r=0}^{\infty} (\alpha^j X)^r.$$

Unfortunately it is not clear what meaning to assign to such a relation. One way round is to work modulo $Z^{2t}$ (more formally, to work in $K[Z]/(Z^{2t})$). We then have $Z^u \equiv 0$ for all integers $u \geq 2t$.

**Lemma 7.20.** *If we work modulo $Z^{2t}$ then*

$$(1-\alpha^j Z)\sum_{m=0}^{2t-1}(\alpha^j Z)^m \equiv 1.$$

Thus, if we work modulo $Z^{2t}$, as we shall from now on, we may define

$$\frac{1}{1-\alpha^j Z} = \sum_{m=0}^{2t-1}(\alpha^j Z)^m.$$

**Lemma 7.21.** *With the conventions already introduced.*

*(i)* $\dfrac{\omega(Z)}{\sigma(Z)} \equiv \displaystyle\sum_{m=0}^{2t-1} Z^m e(\alpha^{m+1})$.

*(ii)* $e(\alpha^m) = r(\alpha^m)$ *for all* $1 \leq m \leq 2t$.

*(iii)* $\dfrac{\omega(Z)}{\sigma(Z)} \equiv \displaystyle\sum_{m=0}^{2t-1} Z^m r(\alpha^{m+1})$.

*(iv)* $\omega(Z) \equiv \sum_{m=0}^{2t-1} Z^m r(\alpha^{m+1})\sigma(Z)$.

*(v)* $\omega_j = \displaystyle\sum_{u+v=j} r(\alpha^{u+1})\sigma_v$ *for all* $0 \leq j \leq 2t-1$.

*(vi)* $0 = \displaystyle\sum_{u+v=j} r(\alpha^{u+1})\sigma_v$ *for all* $t \leq j \leq 2t-1$.

*(vii) The conditions in (vi) determine $\sigma$ completely.*

Part (vi) of Lemma 7.21 completes our search for a decoding method, since $\sigma$ determines $\mathcal{E}$, $\mathcal{E}$ determines $\mathbf{e}$ and $\mathbf{e}$ determines $\mathbf{c}$. It is worth noting that the system of equations in part (v) suffice to determine the pair $\sigma$ and $\omega$ directly.

Compact disk players use BCH codes. Of course errors are likely to occur in bursts (corresponding to scratches etc) and this is dealt with by distributing the bits (digits) in a single codeword over a much longer stretch of track. The code used can correct a burst of 4000 consecutive errors (2.5 mm of track).

Unfortunately none of the codes we have considered work anywhere near the Shannon bound (see Theorem 3.14). We might suspect that this is because they are linear but Elias has shown that this is not the case. (We just state the result without proof.)

**Theorem 7.22.** *In Theorem 3.14 we can replace 'code' by 'linear code'.*

It is clear that much remains to be done.

Just as pure algebra has contributed greatly to the study of error correcting codes so the study of error correcting codes has contributed greatly to the study of pure algebra. The story of one such contribution is set out in T. M. Thompson's *From Error-correcting Codes through Sphere Packings to Simple Groups* [8] — a good, not too mathematical, account of the discovery of the last sporadic simple groups by Conway and others.

## 8 Shift registers

In this section we move towards cryptography but the topic discussed will turn out to have connections with the decoding of BCH codes as well.

**Definition 8.1.** *A* general feedback shift *register is a map* $f : \mathbb{F}_2^d \to \mathbb{F}_2^d$ *given by*

$$f(x_0, x_1, \dots, x_{d-2}, x_{d-1}) = (x_1, x_2, \dots, x_{d-1}, C(x_0, x_1, \dots, x_{d-2}, x_{d-1}))$$

*The* stream *associated to an* initial fill $(y_0, y_1, \dots, y_{d-1})$ *is the sequence*

$$y_0, y_1, \dots, y_j, y_{j+1}, \dots \text{ with } y_n = C(y_{n-d}, y_{n-d+1}, \dots, y_{n-1}) \text{ for all } n \geq d.$$

**Example 8.2.** *If the general feedback shift $f$ given in Definition 8.1 is a permutation then $C$ is linear in the first variable, i.e.*

$$C(x_0, x_1, \dots, x_{d-2}, x_{d-1}) = x_0 + C'(x_1, x_2, \dots, x_{d-2}, x_{d-1}).$$

**Definition 8.3.** *We say that the function $f$ of Definition 8.1 is a* linear feedback register *if*

$$C(x_0, x_1, \dots, x_{d-1}) = a_0 x_{d-1} + a_{d-2} x_1 \cdots + a_0 x_{d-1},$$

*with* $a_{d-1} = 1$.

**Exercise 8.4.** *Discuss briefly the effect of omitting the condition $a_{d-1} = 1$ from Definition 8.3.*

The discussion of the linear recurrence

$$x_n = a_0 x_{n-d} + a_1 x_{n-d-1} \cdots + a_{d-1} x_{n-1}$$

over $\mathbb{F}_2$ follows the 1A discussion of the same problem over $\mathbb{R}$ but is complicated by the fact that

$$n^2 = n$$

in $\mathbb{F}_2$. We assume that $a_0 \neq 0$ and consider the *auxiliary polynomial*

$$C(X) = X^d - a_0X^{d-1} - \cdots - a_{d-2}X - a_{d-1}.$$

In the exercise below $\binom{n}{v}$ is the appropriate polynomial in $n$.

**Exercise 8.5.** *Consider the linear recurrence*

$$x_n = a_0x_{n-d} + a_1x_{n-d-1} \cdots + a_{d-1}x_{n-1} \tag{*}$$

*with $a_j \in \mathbb{F}_2$ and $a_0 \neq 0$.*

*(i) Suppose $K$ is a field containing $\mathbb{F}_2$ such that the auxiliary polynomial $C$ has a root $\alpha$ in $K$. Then $\alpha^n$ is a solution of $(*)$ in $K$.*

*(ii) Suppose $K$ is a field containing $\mathbb{F}_2$ such that the auxiliary polynomial $C$ has $d$ distinct roots $\alpha_1$, $\alpha_2$, ... , $\alpha_d$ in $K$. Then the general solution of $(*)$ in $K$ is*

$$x_n = \sum_{j=1}^{d} b_j\alpha_j^n$$

*for some $b_j \in K$. If $x_0, x_1, \dots, x_{d-1} \in \mathbb{F}_2$ then $x_n \in \mathbb{F}_2$ for all $n$.*

*(iii) Work out the first few lines of Pascal's triangle modulo 2. Show that the functions $f_j : \mathbb{Z} \to \mathbb{F}_2$*

$$f_j(n) = \binom{n}{j}$$

*are linearly independent in the sense that*

$$\sum_{j=0}^{m} a_jf_j(n) = 0$$

*for all $n$ implies $a_j = 0$ for $1 \leq j \leq m$.*

*(iv) Suppose $K$ is a field containing $\mathbb{F}_2$ such that the auxiliary polynomial $C$ factorises completely into linear factors. If the root $\alpha_u$ has multiplicity $m_u$ $[1 \leq u \leq q]$ then the general solution of $(*)$ in $K$ is*

$$x_n = \sum_{u=1}^{q} \sum_{v=0}^{m(u)-1} b_{u,v}\binom{n}{v}\alpha_u^n$$

*for some $b_{u,v} \in K$. If $x_0, x_1, \dots, x_{d-1} \in \mathbb{F}_2$ then $x_n \in \mathbb{F}_2$ for all $n$.*

An strong link with the problem of BCH decoding is provided by Theorem 8.7 below.

**Definition 8.6.** *If we have a sequence (or stream) $x_0, x_1, x_2, \ldots$ of elements of $\mathbb{F}_2$ then its* generating function *$G$ is given by*

$$G(Z) = \sum_{n=0}^{\infty} x_j Z^j$$

**Theorem 8.7.** *The stream $(x_n)$ comes from a linear feedback generator with auxiliary polynomial $C$ if and only if the generating function for the stream is (formally) of the form*

$$G(Z) = \frac{B(Z)}{C(Z)}$$

*with $B$ a polynomial of degree strictly than that of $C$.*

If we can recover $C$ from $G$ then we have recovered the linear feedback generator from the stream.

The link with BCH codes is established by looking at Lemma 7.21 (iii) and making the following remark.

**Lemma 8.8.** *If a stream $(x_n)$ comes from a linear feedback generator with auxiliary polynomial $C$ of degree $d$ then $C$ is determined by the condition*

$$G(Z)C(Z) \equiv B(Z) \mod Z^{2d}$$

*with $B$ a polynomial of degree at most $d-1$.*

We thus have the following problem.
**Problem** *Given a generating function $G$ for a stream and knowing that*

$$G(Z) = \frac{B(Z)}{C(Z)}$$

*with $B$ a polynomial of degree less than that of $C$ and the constant term in $C$ is $c_0 = 1$, recover $C$.*

*The Berlekamp-Massey method* In this method we do not assume that the degree $d$ of $C$ is known. The Berlekamp-Massey solution to this problem is based on the observation that, since

$$\sum_{j=0}^{d} c_j x_{n-j} = 0$$

(with $c_0 = 1$) for all $n \geq d$ we have

$$\begin{pmatrix} x_d & x_{d-1} & \dots & x_1 & x_0 \\ x_{d+1} & x_d & \dots & x_2 & x_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{2d} & x_{2d-1} & \dots & x_{d+1} & x_d \end{pmatrix} \begin{pmatrix} 1 \\ c_1 \\ \vdots \\ c_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \bigstar$$

The Berlekamp-Massey method tells us to look successively at the matrices

$$A_1 = (x_0),\ A_2 = \begin{pmatrix} x_1 & x_0 \\ x_2 & x_1 \end{pmatrix},\ A_3 = \begin{pmatrix} x_2 & x_1 & x_0 \\ x_3 & x_2 & x_1 \\ x_4 & x_3 & x_2 \end{pmatrix}, \dots$$

starting at $A_r$ if it is known that $r \geq d$. For each $A_j$ we evaluate $\det A_j$. If $\det A_j \neq 0$ then $j \neq d$. If $\det A_j = 0$ then $j$ is a good candidate for $d$ so we solve $\bigstar$ on the assumption that $d = j$. (Note that a one dimensional subspace of $\mathbb{F}^{d+1}$ contains only one non-zero vector.) We then check our candidate for $(c_0, c_1, \dots, c_d)$ over as many terms of the stream as we wish. If it fails the test we then know that $d \geq j$ and we start again.

As we have stated it, the Berlekamp-Massey method is not an algorithm in the strict sense of the term although it becomes one if we put an upper bound on the possible values of $d$. (A little thought shows that if no upper bound is put on $d$, no algorithm is possible because, with a suitable initial stream a linear feedback register with large $d$ can be made to produce a stream whose initial values would be produced by a linear feedback register with much smaller $d$. For the same reason the Berlekamp-Massey will produce the $B$ of smallest degree which gives $G$ and not necessarily the original $B$.) In practice, however, the Berlekamp-Massey method is very effective in cases when $d$ is unknown.

It might be thought that evaluating determinants is hard but we can use row reduction to triangularise the matrices and use the fact that $A_{k-1}$ is a sub-matrix of $A_k$ to reduce the work still further.

*A method based on Euclid's algorithm* (This is starred and will be omitted if time is short.) For this method we need to know the degree $d$ of $C$.

Writing $G(Z) = \sum_{j=0}^{\infty} x_j Z^j$ we take $A(Z) = \sum_{j=0}^{2d-1} x_j Z^j$ so that

$$\frac{B(Z)}{C(Z)} = A(Z) + Z^{2d} U(Z)$$

for some power series $U$. It follows that

$$B(Z) = A(Z)C(Z) + Z^{2d} W(Z) \qquad (\dagger)$$

where $A(Z)$ is known but $B(Z)$, $C(Z)$ and the power series $W(Z)$ are unknown.

We now apply Euclid's algorithm to $R_0(Z) = Z^{2d}$, $R_1(Z) = A(Z)$ obtaining, as usual,

$$\begin{aligned} R_0(Z) &= R_1(Z)Q_1(Z) + R_2(Z) \\ R_1(Z) &= R_2(Z)Q_2(Z) + R_3(Z) \\ R_2(Z) &= R_3(Z)Q_3(Z) + R_4(Z) \end{aligned}$$

and so on, but instead of allowing the algorithm to run its full course we stop at the first point when the degree of $R_j$ is no greater than $d$. Call the polynomial $R_j$ so obtained $\tilde{B}$. By the method associated with Bezout's theorem we can find polynomials $\tilde{C}$ and $\tilde{W}$ such that

$$R_j(Z) = R_1(Z)\tilde{C}(Z) + R_0(Z)\tilde{W}(Z)$$

and so

$$\tilde{B}(Z)A(Z)\tilde{C}(Z) + Z^{2d}\tilde{W}(Z). \qquad \dagger\dagger$$

**Lemma 8.9.** *With the notation above.*

*(i) $\tilde{B}$ and $\tilde{C}$ both have degree $d$ or less.*

*(ii) The power series expansions of $\dfrac{B(Z)}{C(Z)}$ and $\dfrac{\tilde{B}(Z)}{\tilde{C}(Z)}$ agree up to the term $Z^{2d}$.*

*(iii) The first $2d$ terms of the power series expansions of $\dfrac{\tilde{B}C - B\tilde{C}}{C\tilde{C}}$ vanish.*

*(iv) The power series of $\dfrac{\tilde{B}C - B\tilde{C}}{C\tilde{C}}$ is the generating sequence for a linear feedback system with auxiliary (or feedback) polynomial $C\tilde{C}$.*

*(v) We have $B = \tilde{B}$ and $C = \tilde{C}$.*

This method is called the Skorobogarov decoder.

## 9 A short homily on cryptography

Cryptography is the science of code making. Cryptanalysis is the art of code breaking.

Two thousand years ago Lucretius wrote that 'Only recently has the true nature of things been discovered'. In the same way mathematicians are apt to feel that 'Only recently has the true nature of cryptography been discovered'. The new mathematical science of cryptography with its promise of codes which are 'provably hard to break' seems to make everything that has gone before irrelevant.

It should, however, be observed that the best cryptographic systems of our ancestors (such as diplomatic 'book codes') served their purpose of ensuring secrecy for a relatively small number of messages between a relatively small number of people extremely well. It is the modern requirement for secrecy *on an industrial scale* to cover endless streams of messages between many centres which has made necessary the modern science of cryptography.

More pertinently it should be remembered that the German Submarine Enigma codes not only appeared to be 'provably hard to break' (though not against the modern criteria of what this should mean) but, *considered in isolation* probably were unbreakable in practice[11]. Fortunately the Submarine codes formed part of an 'Enigma system' with certain exploitable weaknesses. (For an account of how these weaknesses arose and how they were exploited see Kahn's *Seizing the Enigma* [3].)

Even the best codes are like the lock on a safe. However good the lock is, the safe may be broken open by brute force, or stolen together with its comments, or a key holder may be persuaded by fraud or force to open the lock, or the presumed contents of the safe may have been tampered with before they go into the safe, or .... The coding schemes we shall consider, are at best, cryptographic *elements* of larger possible cryptographic *systems*. The planning of cryptographic systems requires not only mathematics but engineering, economics, psychology, humility and an ability to learn from past mistakes. Those who do not learn the lessons of history are condemned to repeat them.

In considering a cryptographic system is important to consider its purpose. Consider a message $M$ sent by $A$ to $B$. Possible aims include

**Secrecy** $A$ and $B$ can be sure that no third party $X$ can read the message $M$.

**Integrity** $A$ and $B$ can be sure that no third party $X$ can alter the message $M$.

**Authenticity** $B$ can be sure that $A$ sent the message $M$.

**Non-repudiation** $B$ can prove to a third party that $A$ sent the message $M$.

When you fill out a cheque giving the sum both in numbers and words you are seeking to protect the *integrity* of the cheque. When you sign a traveller's cheque 'in the presence of the paying officer' the process is intended, from your point of view to protect *authenticity* and, from the bank's point of view to produce *non-repudiation*.

[11] Some versions remained unbroken until the end of the war.

Another point to consider is the level of security aimed at. It hardly matters if a few people use forged tickets to travel on the underground, it does matter if a single unauthorised individual can gain privileged access to a bank's central computer system. If *secrecy* is aimed at, how long must the secret be kept? Some military and financial secrets need only remain secret for a few hours, others must remain secret for years.

We must also, to conclude this non-exhaustive list, consider the level of security required. Here are three possible levels.

(1) Prospective opponents should find it hard to compromise your system even if they are in possession of a plentiful supply of encoded messages $C_i$.

(2) Prospective opponents should find it hard to compromise your system even if they are in possession of a plentiful supply of pairs $(M_i, C_i)$ of messages $M_i$ together with their encodings $C_i$.

(3) Prospective opponents should find it hard to compromise your system even if they are allowed to produce messages $M_i$ and given their encodings $C_i$.

Clearly safety at level (3) implies safety at level (2) and safety at level (2) implies safety at level (1). Roughly speaking, the best Enigma codes satisfied (1). The German Navy believed on good but mistaken grounds that they satisfied (2). Level (3) would have appeared evidently impossible to attain until a few years ago. Nowadays, level (3) is considered a minimal requirement for a really secure system.

## 10 Stream cyphers

One natural way of enciphering is to use a *stream cypher*. We work with streams (that is, sequences) of elements of $\mathbb{F}_2$. We use *cypher stream* $k_0$, $k_1$, $k_2$ .... The *plain text stream* $p_0$, $p_1$, $p_2$, ... is enciphered as the *cypher text stream* $z_0$, $z_1$, $z_2$, ... given by

$$z_n = p_n + k_n.$$

This is an example of a *private key* or *symmetric* system. The security of the system depends on a secret (in our case the cypher stream) $\mathbf{k}$ shared between the cypherer and the encipherer. Knowledge of an enciphering method makes it easy to work out a deciphering method and vice versa. In our case a deciphering method is given by the observation that

$$p_n = z_n + k_n.$$

(Indeed, writing $\alpha(\mathbf{p}) = \mathbf{p} + \mathbf{z}$ we see that the enciphering function $\alpha$ has the property that $\alpha^2 = \iota$ the identity map. Cyphers like this are called *symmetric*.)

In the one-time pad first discussed by Vernam in 1926 the cypher stream is a random sequence $k_j = K_j$ where the $K_j$ are independent random variables with

$$\Pr(K_j = 0) = \Pr(K_j = 1) = 1/2.$$

If we write $Z_j = p_j + K_j$ then we see that the $P_j$ are independent random variables with

$$\Pr(P_j = 0) = \Pr(P_j = 1) = 1/2.$$

Thus (in the absence of any knowledge of the ciphering stream) the code-breaker is just faced by a stream of perfectly random binary digits. Decipherment is impossible in principle.

It is sometimes said that it is hard to find random sequences, and it is indeed rather harder than might appear at first sight, but it is not too difficult to rig up a system for producing 'sufficiently random' sequences[12]. The secret services of the former Soviet Union were particularly fond of one-time pads. The real difficulty lies in the necessity for sharing the secret sequence $\mathbf{k}$. If a random sequence is reused it ceases to be random (it becomes 'the same code as last Wednesday' or the 'the same code as Paris uses') so, when there is a great deal of code traffic, new one-time pads must be sent out. If random bits can be safely communicated so can ordinary messages and the exercise becomes pointless.

In practice we would like to start from a short shared secret 'seed' and generate a ciphering string $\mathbf{k}$ that 'behaves like a random sequence'. This leads us straight into deep philosophical waters[13]. As might be expected there is an illuminating discussion in Chapter III of Knuth's marvellous *The Art of Computing Programming* [6]. Note in particular his warning:

> ... *random numbers should not be generated with a method chosen at random.* Some theory should be used.

One way that we might try to generate our ciphering string is to use a general feedback shift register $f$ of length $d$ with the initial fill $(k_0, k_1, \dots, k_{d-1})$ as the secret seed.

[12]Take ten of your favourite long books, convert to binary sequences $x_{j,n}$ and set $k_n = \sum_{j=1}^{10} x_{j,1000+j+n} + s_n$ where $s_n$ is the output of your favourite 'pseudo-random number generator'. Give a disc with a copy of $\mathbf{k}$ to your friend and, provided both of you obey some elementary rules, your correspondence will be safe from MI5. The anguished debate in the US about codes and privacy refers to the privacy of large organisations and their clients, not the privacy of communication from individual to individual.

[13]Where we drown at once, since, the best (at least my opinion) modern view is that any sequence that can be generated by a program of reasonable length from a 'seed' of reasonable size is automatically non-random.

**Lemma 10.1.** *If $f$ is a general feedback shift register of length $d$ then given any initial fill $(k_0, k_1, \ldots, k_{d-1})$ there will exist $N, M \leq 2^d$ such that the output stream $\mathbf{k}$ satisfies $k_{r+N} = k_r$ for all $r \geq M$.*

**Lemma 10.2.** *Suppose that $f$ is a linear feedback register of length $d$.*

*(i) $f(x_0, x_1, \ldots, x_{d-1}) = (x_0, x_1, \ldots, x_{d-1})$ if and only if $(x_0, x_1, \ldots, x_{d-1}) = (0, 0, \ldots, 0)$.*

*(ii) Given any initial fill $(k_0, k_1, \ldots, k_{d-1})$ there will exist $N, M \leq 2^d - 1$ such that the output stream $\mathbf{k}$ satisfies $k_{r+N} = k_r$ for all $r \geq M$.*

We can complement Lemma 10.2 by using Lemma 6.15 and the associated the discussion.

**Lemma 10.3.** *A linear feedback register of length $d$ attains its maximal period $2^d - 1$ (for a non-trivial initial fill) when the roots of the feedback polynomial are primitive elements of $\mathbb{F}^{2^d}$.*

(We will note why this result is plausible but we will not prove it.)

It is well known that short period streams are dangerous. During World War II the British Navy used codes whose period was adequately long for peace time use. The massive increase in traffic required by war time conditions meant that the period was now too short. By dint of immense toil German naval code breakers were able to identify coincidences and by this means slowly break the British codes.

Unfortunately, whilst short periods are definitely unsafe it does not follow that long periods guarantee safety. Using the Berlekamp-Massey method we see that stream codes based on linear feedback registers are unsafe at level (2).

**Lemma 10.4.** *Suppose that an unknown* cypher stream *$k_0$, $k_1$, $k_2$ $\ldots$ is produced by an unknown linear feedback register $f$ of unknown length $d \leq D$. The* plain text stream *$p_0$, $p_1$, $p_2$, $\ldots$ is enciphered as the* cypher text stream *$z_0$, $z_1$, $z_2$, $\ldots$ given by*

$$z_n = p_n + k_n.$$

*If we are given $p_0$, $p_1$, $\ldots$ $p_{2D-1}$ and $z_0$, $z_1$, $\ldots$ $z_{2D-1}$ then we can find $k_r$ for all $r$.*

Thus if we have a message of length twice the length of the linear feedback register together with its encipherment the code is broken.

It is easy to construct immensely complicated looking linear feedback registers with hundreds of registers. Lemma 10.4 shows that, from the point of view of a determined, well equipped and technically competent opponent, cryptographic systems based on such registers are the equivalent of leaving your house key hidden under the door mat. Professionals say that such systems seek ‘security by obscurity’.

However, if you do not wish to baffle the CIA, but merely prevent little old ladies in tennis shoes watching subscription television without paying for it, systems based on linear feedback registers are cheap and quite effective. Whatever they may say in public, large companies are happy to tolerate a certain level of fraud. So long as 99.9% of the calls made are paid for, the profits of a telephone company are essentially unaffected by the .1% which ‘break the system’.

What happens if we try some simple tricks to increase the complexity of the cypher text stream.

**Lemma 10.5.** *If $x_n$ is a stream produced by a linear feedback system of length $N$ with auxiliary polynomial $P$ and $y_n$ is a stream produced by a linear feedback system of length $N$ with auxiliary polynomial $Q$ then $x_n + y_n$ is a stream produced by a linear feedback system of length $N + M$ with auxiliary polynomial $P(X)Q(X)$.*

Note that this means that adding streams from two linear feedback system is no more economical than producing the same effect with one. Indeed the situation may be worse since *a stream produced by linear feedback system of given length may, possibly, also be produced by another linear feedback system of shorter length.*

**Lemma 10.6.** *Suppose that $x_n$ is a stream produced by a linear feedback system of length $N$ with auxiliary polynomial $P$ and $y_n$ is a stream produced by a linear feedback system of length $N$ with auxiliary polynomial $Q$. Let $P$ have roots $\alpha_1$, $\alpha_2$, … $\alpha_N$ and $Q$ have roots $\beta_1$, $\beta_2$, … $\beta_M$ over some field $K \supseteq \mathbb{F}_2$. Then $x_n y_n$ is a stream produced by a linear feedback system of length $NM$ with auxiliary polynomial*

$$\prod_{1\leq i\leq N}\prod_{1\leq i\leq M}(X-\alpha_i\beta_j).$$

We shall probably only prove Lemmas 10.5 and 10.6 in the case when all roots are distinct, leaving the more general case as an easy exercise. We shall also not prove that the polynomial $\prod_{1\leq i\leq N}\prod_{1\leq i\leq M}(X-\alpha_i\beta_j)$ obtained in Lemma 10.6 actually lies in $\mathbb{F}_2$ but (for those who are familiar with the phrase in quotes) this is an easy exercise in ‘symmetric functions of roots’.

Here is an even easier remark.

**Lemma 10.7.** *Suppose that $x_n$ is a stream which is periodic with period $N$ and $y_n$ is a stream which is periodic with period $M$. Then the streams $x_n + y_n$ and $x_n y_n$ are periodic with periods dividing the lowest common multiple of $N$ and $M$.*

**Exercise 10.8.** *One of the most confidential German codes (called FISH by the British) involved a complex mechanism which the British found could be simulated by two loops of paper tape of length* 1501 *and* 1497*. If $k_n = x_n + y_n$ where $x_n$ is a stream of period* 1501 *and $y_n$ is stream of period* 1497 *what is the longest possible period of $k_n$. How many consecutive values of $k_n$ do you need to to specify the sequence completely.*

It might be thought that the lengthening of the underlying linear feedback system obtained in Lemma 10.6 is worth having but it is bought at a substantial price. Let me illustrate this by an informal argument. Suppose we have 10 streams $x_{j,n}$ (without any peculiar properties) produced linear feedback registers of length about 100. If we form $k_n = \prod_{j=1}^{10} x_{j,n}$ then the Berlekamp-Massey method requires of the order of $10^{20}$ consecutive values of $k_n$ and the periodicity of $k_n$ can be made still more astronomical. Our cypher key stream $k_n$ appears safe from prying eyes. However it is doubtful if the prying eyes will mind. Observe that (under reasonable conditions) about $2^{-1}$ of the $x_{j,n}$ will have the value 1 and about $2^{-10}$ of the $k_n = \prod_{j=1}^{10} x_{j,n}$ will have value 1. Thus if $z_n = p_n + k_n$, in more than 999 cases out of a 1000 we will have $z_n = p_n$. Even if we just combine two streams $x_n$ and $y_n$ in the way suggested we may expect $x_n y_n = 0$ for about 75% of the time.

Here is another example where the apparent complexity of the cypher key stream is substantially greater than its true complexity.

**Example 10.9.** *The following is a simplified version of a standard satellite TV decoder. We have 3 streams $x_n$, $y_n$, $z_n$ produced by linear feedback registers. If the cypher key stream is defined by*

$$\begin{aligned} k_n &= x_n \qquad \textit{if } z_n = 0 \\ k_n &= y_n \qquad \textit{if } z_n = 1 \end{aligned}$$

*then*

$$k_n = (y_n + x_n) z_n + x_n$$

*and the cypher key stream is that produced by linear feedback register.*

It might be thought that the best way round these difficulties is to use a non-linear feedback generator $f$. This is not the easy way out that it appears.

If chosen by an amateur the complicated looking $f$ so produced will have the apparent advantage that we do not know what is wrong with it and the very real disadvantage that we do not know what is wrong with it.

Another approach is to observe that, so far as the potential code breaker is concerned, the cypher stream method only combines the 'unknown secret' (here the feedback generator $f$ together with the seed $(k_0, k_1, \dots, k_{d-1})$) with the unknown message $\mathbf{p}$ in a rather simple way. It might be better to consider a system with two functions $F : \mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{F}_2^q$ and $G : \mathbb{F}_2^m \times \mathbb{F}_2^q \to \mathbb{F}_2^n$. such that

$$G(\mathbf{k}, F(\mathbf{k}, \mathbf{p})) = \mathbf{p}.$$

Here $\mathbf{k}$ will be the shared secret, $\mathbf{p}$ the message $\mathbf{z} = F(\mathbf{k}, \mathbf{p})$ the encoded message we can be decoded by using the fact that $G(\mathbf{k}, \mathbf{z}) = \mathbf{p}$.

In the next section we shall see that an even better arrangement is possible. However, arrangements like this have the disadvantage that the the message $\mathbf{p}$ must be entirely known before it is transmitted and the encoded message $\mathbf{z}$ must have been entirely received before in can be decoded. Stream ciphers have the advantage that they can be decoded 'on the fly'. They are also much more error tolerant. A mistake in the coding, transmission or decoding of a single element only produces an error in a single place of the sequence. There will continue to be circumstances where stream ciphers are appropriate.

There is one further remark to be made. Suppose that, as is often the case, that we know $F$, that $n = q$ and we know the 'encoded message' $\mathbf{z}$. Suppose also that we know that the 'unknown secret' or 'key' $\mathbf{k} \in \mathcal{K} \subseteq \mathbb{F}_2^m$ and the 'unknown message' $\mathbf{p} \in \mathcal{P} \subseteq \mathbb{F}_2^n$. We are then faced with the problem:- Solve the system

$$\mathbf{z} = F(\mathbf{k}, \mathbf{p}) \text{ where } \mathbf{k} \in \mathcal{K},\ \mathbf{p} \in \mathcal{P}. \qquad \bigstar$$

Speaking roughly, the task is hopeless unless $\bigstar$ has a unique solution[14]. Speaking even more roughly, this is unlikely to happen if $|\mathcal{K}||\mathcal{P}| > 2^n$ and is likely to happen if $2^n$ is substantially greater than $|\mathcal{K}||\mathcal{P}|$. (Here, as usual, $|\mathcal{B}|$ denotes the number of elements of $\mathcal{B}$.)

[14]'According to some, the primordial Torah was inscribed in black flames on white fire. At the moment of its creation, it appeared as a series of letters not yet joined up in the form of words. For this reason, in the Torah rolls there appear neither vowels nor punctuation, nor accents; for the original Torah was nothing but a disordered heap of letters. Furthermore, had it not been for Adam's sin, these letters might have been joined differently to form another story. For the kabalist, God will abolish the present ordering of the letters, or else will teach us how to read them according to a new disposition only after the coming of the Messiah.' ([1], Chapter 2.)

Now recall the definition of the information rate given in Definition 1.2. If the message set $\mathcal{M}$ has information rate $\mu$ and the key set (that is the shared secret set) $\mathcal{K}$ has information rate $\kappa$ then, taking logarithms we see that if

$$n - m\kappa - n\mu$$

is substantially greater than 0 then $\bigstar$ is likely to have a unique solution, but if it is substantially smaller this is unlikely.

**Example 10.10.** *If instead of using binary code we consider an alphabet of 27 letters (the English alphabet plus a space) we must take logarithms to the base 27 but the considerations above continue to apply. The English language treated in this way has information rate about .4. (This is very much a ball park figure. The information rate is certainly less than .5 and almost certainly greater than .2.)*

*(i) In the Caesar code we replace the ith element of our alphabet by the $i + j$th (modulo 27). The shared secret is a single letter (the code for A say). We have $m = 1$, $\kappa = 1$ and $\mu \approx .4$.*

$$n - m\kappa - n\mu \approx .6n - 1.$$

*If $n = 1$ (so $n - m\kappa - n\mu \approx -.4$) it is obviously impossible to decode the message. If $n = 10$ (so $n - m\kappa - n\mu \approx 5$) a simple search through the 27 possibilities will almost always give a single possible decode.*

*(ii) A simple substitution code a permutation of the alphabet is chosen and applied to each letter of the code in turn. The shared secret is a sequence of 26 letters (giving the coding of the first 26 letters, the 27th can then be deduced). We have $m = 26$, $\kappa = 1$ and $\mu \approx .4$.*

$$n - m\kappa - n\mu \approx .6n - 26.$$

*In* the Dancing Men *Sherlock Holmes solves such a code with $n = 68$ (so $n - m\kappa - n\mu \approx 15$) without straining the reader's credulity too much and would think that, unless the message is very carefully chosen most of my audience could solve such a code with $n = 200$ (so $n - m\kappa - n\mu \approx 100$).*

*(iii) In the one-time pad $m = n$ and $\kappa = 1$ so (if $\mu > 0$)*

$$n - m\kappa - n\mu = -n\mu \to -\infty$$

*as $n \to \infty$.*

*(iv) Note that the larger $\mu$ is the slower $n - m\kappa - n\mu$ increases. This corresponds to the very general statement that the higher the information rate of the messages the harder it is to break the code in which they are sent.*

The ideas just introduced can be formalised by the notion of unicity distance.

**Definition 10.11.** *The* unicity distance *of a code is the number of bits of message required to exceed the number of bits of information in the key plus the number of bits of information in the message.*

If the reader complains that there is a faint smell of red herring about this definition, I would be inclined to agree. Without a clearer discussion of 'information content' than is given in this course it must remain more of a slogan than a definition.

If we only use our code once to send a message which is substantially shorter than the unicity distance we can be confident that no code breaker, however gifted, could break it, simply because there is there is no unambiguous decode. (A one-time pad has unicity distance infinity.) However, the fact that there is a unique solution to a problem does not mean that it is easy to find. We have excellent reasons, some of which are spelled out in the next section, to believe that there exist codes for which the unicity distance is essentially irrelevant to the maximum safe length of a message.

## 11 Asymmetric systems

Towards the end of the previous section we discussed a general coding scheme depending on a shared secret key $\mathbf{k}$ known to the encoder and the decoder. However, the scheme can be generalised still further by splitting the secret in two. Consider a system with two functions $F : \mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{F}_2^q$ and $G : \mathbb{F}_2^m \times \mathbb{F}_2^p \to \mathbb{F}_2^n$. such that

$$G(\mathbf{l}, F(\mathbf{k}, \mathbf{p})) = \mathbf{p}.$$

Here $(\mathbf{k}, \mathbf{l})$ will be be a pair of secrets, $\mathbf{p}$ the message $\mathbf{z} = F(\mathbf{k}, \mathbf{p})$ the encoded message which can be decoded by using the fact that $G(\mathbf{l}, \mathbf{z}) = \mathbf{p}$. In this scheme the encoder must know $\mathbf{k}$ but need not know $\mathbf{l}$ and the decoder must know $\mathbf{l}$ and but need not know $\mathbf{k}$. Such a system is called assymetric.

So far the idea is interesting but not exciting. Suppose however, that we can show that

(i) knowing $F$, $G$ and $\mathbf{l}$ it is very hard to find $\mathbf{k}$,

(ii) if we do not know $\mathbf{k}$ then, even if we know $F$ and $G$, it very hard to find $\mathbf{p}$ $F(\mathbf{k}, \mathbf{p})$.

Then the code is secure at what we called level (3).

**Lemma 11.1.** *Suppose that the conditions specified above hold. Then an opponent who is entitled to demand the encodings $\mathbf{z}_i$ of any messages $\mathbf{p}_i$ they choose to specify will still find it very hard to find $\mathbf{p}$ when given $F(\mathbf{k}, \mathbf{p})$.*

Let us write $F(\mathbf{k}, \mathbf{p}) = \mathbf{p}^{K_A}$ and $G(\mathbf{l}, \mathbf{z}) = \mathbf{z}^{K_A^{-1}}$ and think of $\mathbf{p}^{K_A}$ as participant $A$'s encipherment of $\mathbf{p}$ and $\mathbf{z}^{K_A^{-1}}$ as participant $B$'s decipherment of $\mathbf{z}$. We then have

$$(\mathbf{p}^{K_A})^{K_A^{-1}} = \mathbf{p}.$$

Lemma 11.1 tells us that such a system is secure however many messages are sent. Moreover, if we think of $A$ a a spy-master he can broadcast $K_A$ to the world (that is why such systems are called public key systems) and invite anybody who wants to spy for him to send him secret messages in total confidence.

It is all very well to describe such a code but do they exist? There is very strong evidence that they do but so far all mathematicians have been able to do is to show that provided certain mathematical problems which are believed to be hard are indeed hard then good codes exist.

The following problem is believed to be hard.

**Problem** *Given an integer $N$ which is known to be the product $N = pq$ of two primes $p$ and $q$, find $p$ and $q$.*

Several schemes have been proposed based on assumption that this factorisation is hard. (Note however that it is easy to find large primes $p$ and $q$.) We give a very elegant scheme due to Rabin and Williams. It makes use of some simple number theoretic results from 1A and 1B.

The following result was proved towards the end of the course *Quadratic Mathematics* and is, in any case, easy to obtain by considering primitive roots.

**Lemma 11.2.** *If $p$ is an odd prime the congruence*

$$x^2 \equiv d \mod p$$

*is soluble if and only if $d \equiv 0$ or $d^{(p-1)/2} \equiv 1$ modulo $p$.*

**Lemma 11.3.** *Suppose $p$ is a prime such that $p = 4k - 1$ for some integer $k$. Then if the congruence*

$$x^2 \equiv d \mod p$$

*has any solution, it has $d^k$ as a solution.*

We now call on the Chinese remainder theorem.

**Lemma 11.4.** *Let $p$ and $q$ be primes of the form $4k-1$ and set $N = pq$. Then the following two problems are of equivalent difficulty.*

*(A) Given $N$ and $d$ find all the $m$ satisfying*

$$m^2 \equiv d \mod N.$$

*(B) Given $N$ find $p$ and $q$.*

(Note that, provided that that $d \not\equiv 0$, knowing the solution to (A) for any $d$ gives us the four solutions for $d = 1$.) The result is also true but much harder to prove for general primes $p$ and $q$.

At the risk of giving aid and comfort to followers of the Lakatosian heresy it must be admitted that the statement of Lemma 11.4 does not really tell us what the result we are proving is, although the proof makes it clear that the result (whatever it may be) is certainly true. However, with more work, everything can be made precise.

We can now give the Rabin-Williams scheme. The spy-master $A$ selects two very large primes $p$ and $q$. (Since he has only done an undergraduate course in mathematics he will take $p$ and $q$ of the form $4k-1$.) He keeps the pair $(p, q)$ secret but broadcasts the public key $N = pq$. If $B$ wants to send him a message she writes it in binary code splits it into blocks of length $m$ with $2^m < N < 2^{m+1}$. Each of these blocks is a number $r_j$ with $0 \leq r_j < N$. $B$ computes $s_j$ such that $r_j^2 \equiv s_j$ modulo $N$ and sends $s_j$. The spy-master (who knows $p$ and $q$) can use the method of Lemma 11.4 to find one of four possible values for $r_j$ (the four square roots of $s_j$). Of these four possible message blocks it is almost certain that three will be garbage so the fourth will be the desired message.

If the reader reflects, she will see that the ambiguity of the root is genuinely unproblematic. (If the decoding is mechanical then making each block start with some fixed sequence of length 50 will reduce the risk of ambiguity to negligible proportions.) Slightly more problematic, from the practical point of view, is the possibility that some one could be known to have sent a very short message, that is to have started with an $m$ such that $1 \leq m \leq N^{1/2}$ but provided sensible precautions are taken this should not occur.

## 12 Commutative public key systems

In the previous sections we introduced the coding and decoding functions $K_A$ and $K_A^{-1}$ with the property that

$$(\mathbf{p}^{K_A})^{K_A^{-1}} = \mathbf{p},$$

and satisfying the condition that knowledge of $K_A$ did not help very much in finding $K_A^{-1}$. We usually require, in addition, that our system be *commutative* in the sense that

$$(\mathbf{p}^{K_A^{-1}})^{K_A} = \mathbf{p}.$$

and that knowledge of $K_A^{-1}$ does not help very much in finding $K_A$. The Rabin–Williams scheme, as described in the last section, does not have this property.

Commutative public key codes are very flexible and provide us with simple means for maintaining integrity, authenticity and non-repudiation. (This is not to say that non-commutative codes can not do the same; simply that commutativity makes many things easier.)

**Integrity and non-repudiation** Let $A$ 'own a code', that is know both $K_A$ and $K_A^{-1}$. Then $A$ can broadcast $K_A^{-1}$ to everybody so that everybody can decode but only $A$ can encode. (We say that $K_A^{-1}$ is the *public key* and $K_A$ the *private key*.) Then, for example, example, $A$ could issue tickets to the castle ball carrying the coded message 'admit Joe Bloggs' which could be read by the recipients and the guards but would be unforgeable. However, for the same reason, $A$ could not deny that he had issued the invitation.

**Authenticity** If $B$ wants to be sure that $A$ is sending a message then $B$ can send $A$ a harmless random message $\mathbf{q}$. If $B$ receives back a message $\mathbf{p}$ such that $\mathbf{p}^{K_A^{-1}}$ ends with the message $\mathbf{q}$ then $A$ must have sent it to $B$. (Any body can *copy* a coded message but only $A$ can control the content.)

**Signature** Suppose now that $B$ owns a commutative code pair $(K_B, K_B^{-1})$ and has broadcast $K_B^{-1}$. If $A$ wants to send a message $\mathbf{p}$ to $B$ he computes $\mathbf{q} = \mathbf{p}^{K_A}$ and sends $\mathbf{p}^{K_B^{-1}}$ followed by $(\mathbf{q}^{K_A^{-1}})^{K_B^{-1}}$. $B$ can now use the fact that

$$(\mathbf{q}^{K_B^{-1}})^{K_B} = \mathbf{q}$$

to recover $\mathbf{p}$ and $\mathbf{q}$. $B$ then observes that $\mathbf{q}^{K_A^{-1}} = \mathbf{p}$. Since only $A$ can produce a pair $(\mathbf{p}, \mathbf{q})$ with this property, $A$ must have written it.

There is now a charming little branch of the mathematical literature based on these ideas in which Albert gets Bertha to authenticate a message from Caroline to David using information from Eveline and Fitzpatrick, Gilbert and Harriet play coin tossing down the phone and Ingred, Jacob, Katherine and Laszlo play bridge without using a pack of cards. However a cryptographic system is only as strong as its weakest link. Unbreakable password systems do not prevent computer systems being regularly penetrated by 'hackers' and however 'secure' a transaction on the net may be it may still involve a rogue on one end and a fool on the other.

The most famous candidate for a commutative public key system is the RSA (Rivest, Shamir, Adleman) system. It was the RSA system the first convinced the mathematical community that public key systems might be feasible. The reader will have met the RSA in 1A but we will push the ideas a little bit further.

**Lemma 12.1.** *Let $p$ and $q$ be primes. If $N = pq$ and $\lambda(N) = \operatorname{lcm}(p-1, q-1)$ then*

$$M^{\lambda(N)} \equiv 1 \pmod{N}$$

*for all integers $M$.*

Since we wish to appeal to Lemma 11.4 we shall assume in what follows that we have secretly chosen large primes $p$ and $q$ of the form $4k-1$. (However, as before, the arguments can be made to work for general large primes $p$ and $q$.) We choose an integer $e$ and then use Euclid's algorithm to find an integer $d$ such that

$$de \equiv 1. \pmod{\lambda(N)}$$

Since others may be better psychologists than we are, we would be wise to use some sort of random method for choosing $p$, $q$ and $e$.

The public key includes the value of $d$ and $N$ but we keep secret the value of $e$. Given a number $M$ with $1 \leq M \leq N-1$ we encode it as the integer $E$ with $1 \leq M \leq N-1$

$$E \equiv M^d \pmod{N}.$$

The *public* decoding method is given by the observation that

$$E^e \equiv M^{de} \equiv M.$$

As was observed in 1A, high powers are easy to compute.

To show that (providing that factoring $N$ is indeed hard) finding $d$ from $e$ and $N$ is hard we use the following lemma.

**Lemma 12.2.** *Suppose that $d$, $e$ and $N$ are as above. Set $de - 1 = 2^a b$ where $b$ is odd.*

*(i) $a \geq 1$.*

*(ii) If $y \equiv x^b \pmod{N}$ then there exists an $r$ with $1 \leq r \leq 2^{a-1}$ such that*

$$y^r \not\equiv 1 \textit{ but } y^r \equiv 1 \pmod{N}.$$

Combined with Lemma 11.4, the idea of Lemma 12.2 gives a fast *probabilistic algorithm* where by making random choices of $x$ we very rapidly reduce the probability that we can not find $p$ and $q$ to as close to zero as we wish.

**Lemma 12.3.** *The problem of finding $d$ from the public information $e$ and $N$ is, essentially as hard as factorising $N$.*

*Remark 1* At first glance we seem to have done as well for the RSA code as for the Rabin–Williams code. But this is not so. In Lemma 11.4 we showed that finding the four solutions of $M^2 \equiv E \pmod{N}$ was equivalent to factorising $N$. In the absence of further information, finding one root is as hard as finding another. Thus the ability to break the Rabin-Williams code (without some tremendous stroke of luck) is equivalent to the ability to factor $N$. On the other hand it is a priori, possible that it might be possible to find a decoding method for the RSA code which did not involve knowing $d$. Thus it might be possible to break the RSA code without finding $d$. It must, however, be said that, in spite of this problem, the RSA code is much used in practice and the Rabin–Williams code is not.

*Remark 2* It is natural to ask what evidence there is that the factorisation problem really is hard. Properly organised, trial division requires $O(N^{1/2})$ operations to factorise a number $N$. This order of magnitude was not bettered until 1972 when Lehman produced a $O(N^{1/3})$ method. In 1974, Pollard[15] produced a $O(N^{1/4})$ method. In 1979, as interest in the problem grew because of its connection with secret codes, Lenstra made a breakthrough to a $O(e^{c((\log N)(\log\log N))^{1/2}})$ method with $c \approx 2$. Since then some progress has been made (Pollard reached $O(e^{2((\log N)(\log\log N))^{1/3}})$ but in spite of intense efforts mathematicians have not produced anything which would be a real threat to codes based on the factorisation problem. In 1996, it was possible to factor 100 (decimal) digit numbers routinely, 150 digit numbers with immense effort but 200 digit numbers were out of reach.

Organisations which use the RSA and related systems rely on 'security through publicity'. Because the problem of cracking RSA codes is so notorious any breakthrough is likely to be publically announced[16]. Moreover, even if a breakthrough occurs it is unlikely to be one which can be easily exploited by the average criminal. So long as the secrets covered by RSA-type codes need only be kept for a few months rather than forever, the codes can be considered to be one of the strongest links in the security chain.

[15] Although mathematically trained, Pollard worked outside the professional mathematical community.

[16] And if not, is most likely to be a government rather than a Mafia secret.

## 13 Trapdoors and signatures

It might be thought that secure codes are all that are needed to ensure the security of communications but this is not so. It is not necessary to read a message to derive information from it[17]. In the same way, it may not be necessary to be able to write a message in order to tamper with it.

Here is a somewhat far fetched but worrying example. Suppose that by wire tapping or by looking over peoples' shoulders I find that a bank creates messages in the form $M_1$, $M_2$ where $M_1$ is the name of the client and $M_2$ is the sum to be transfered to the client's account. The messages are then encoded according to the RSA scheme discussed after Lemma 12.1 as $Z_1 = M_1^d$ and $Z_2 = M_2^d$. I then enter into a transaction with the bank which adds \$ 1000 to my account. I observe the resulting $Z_1$ and $Z_2$ and the transmit $Z_1$ followed by $Z_2^3$.

**Example 13.1.** *What will (I hope) be the result of this transaction.*

We say that the RSA scheme is vulnerable to 'homomorphism attack'.

One way of increasing security against tampering is to first code our message by classical coding method and then use our RSA (or similar) scheme on the result.

**Exercise 13.2.** *Discuss briefly the effect of first using an RSA scheme and then a classical code.*

However there is another way forward which has the advantage of wider applicability since it also can be used to protect the integrity of open (non-coded) messages and to produce password systems. These are the so called *signature systems.* (Note that we shall be concerned with the 'signature of the message' and not the signature of the sender.)

**Definition 13.3.** *A* signature *or* trapdoor *or* hashing *function is a mapping $H : \mathcal{M} \to \mathcal{S}$ from the space $\mathcal{M}$ of possible messages to the space $\mathcal{S}$ of possible signatures.*

(Let me admit at once that Definition 13.3 is more of a statement of notation than a useful definition.) The first requirement of a good signature function is that the space $\mathcal{M}$ should be much larger than the space $\mathcal{S}$ so that $H$ is a many-to-one function (in fact a great-many-to-one function) so that we can not work back from $H(M)$ to $M$. The second requirement is that $\mathcal{S}$ should be large so that a forger can not (sensibly) hope to hit on $H(M)$ by luck.

[17]During World War II, British bomber crews used to spend the morning before a night raid testing their equipment, this included the radios.

Obviously we should aim at the same kind of security as that offered by our 'level 2' for codes:-

> Prospective opponents should find it hard to find $H(M)$ given $M$ if they are in possession of a plentiful supply of message, signature pairs $(M_i, H(M_i))$. of messages $M_i$ together with their encodings $C_i$.

I leave it to the reader to think about level 3 security (or to look at section 12.6 of [9]).

Here is a signature scheme due to Elgamal. The message sender $A$ chooses a very large prime $p$, some integer $1 < g < p$. and some other integer $u$ with $1 < u < p$ (as usual, some randomisation scheme should be used). $A$ then releases the values of $p$, $g$ and $g^u$ (modulo $p$) but keeps the value of $u$ secret. Whenever he sends a message $m$ (some positive integer) he chooses another integer $k$ with $1 \leq k \leq p-2$ at random and computes $r$ and $s$ with $1 \leq r \leq p-1$ and $0 \leq s \leq p-2$ by the rules[18]

$$r \equiv g^k \pmod{p} \tag{*}$$

$$m \equiv ur + ks \pmod{p-1} \tag{**}$$

**Lemma 13.4.** *If conditions (*) and (**) are satisfied then*

$$g^m \equiv y^r r^s \pmod{p}$$

If $A$ sends the message $m$ followed by the signature $(r, s)$ the recipient need only verify the relation $g^m \equiv y^r r^s \pmod{p}$ to check that the message is authentic.

Since $k$ is random it is *believed* that the only way to forge signatures is to find $u$ from $g^u$ and it is *believed* that this problem, which is known as the discrete logarithm problem is very hard.

Needless to say, even if it is impossible to tamper with a message, signature pair it is always possible to copy one. Every message should thus contain a unique identifier such as a time stamp.

The evidence that the discrete logarithm problem is very hard is of the same kind of nature and strength as the evidence that the factorisation problem is very hard. We conclude our discussion with a description of the Diffie–Helman key exchange system which is also based on the discrete logarithm problem.

[18]There is a small point which I have glossed over here and elsewhere. Unless $k$ and and $p-1$ are coprime the equation (**) may not be soluble. However the quickest way to solve (**) if it is soluble is Euclid's algorithm which will also reveal if (**) is insoluble. If (**) is insoluble we simply choose another $k$ at random and try again.

The modern coding schemes which we have discussed have the disadvantage that they require lots of computation. This is not a disadvantage when we deal slowly with a few important messages. For the Web where we must deal speedily with a lot of less than world shattering messages sent by impatient individuals this is a grave disadvantage. Classical coding schemes are fast but become insecure with reuse. *Key exchange schemes* use modern codes to communicate a new secret key for each message. Once the secret key has been sent slowly, a fast classical method based on the secret key is used to encode and decode the message. Since a different secret key is used each time, the classical code is secure.

How is this done? Suppose $A$ and $B$ are at opposite ends of a tapped telephone line. $A$ sends $B$ a (randomly chosen) large prime $p$ and a randomly chosen $g$ with $1 < g < p-1$. Since the telephone line is insecure $A$ and $B$ must assume that $p$ and $g$ are public knowledge. $A$ now chooses randomly a secret number $\alpha$ and tells $B$ the value of $g^\alpha$. $B$ chooses randomly a secret number $\beta$ and tells $A$ the value of $g^\beta$. Since

$$g^{\alpha\beta} = (g^\alpha)^\beta = (g^\beta)^\alpha,$$

both $A$ and $B$ can compute $k = g^{\alpha\beta}$ modulo $p$ and $k$ becomes the shared secret key.

The eavesdropper is left with the problem of finding $k \equiv g^{\alpha\beta}$ from knowledge of $g$, $g^\alpha$ and $g^\beta$ (modulo $p$). It is conjectured that this is essentially as hard as finding $\alpha$ and $\beta$ from the values of $g$, $g^\alpha$ and $g^\beta$ (modulo $p$) and this is the discrete logarithm problem.

We conclude with a quotation from Galbraith (referring to his time as ambassador to India) taken from Koblitz's entertaining text [5].

> I had asked that a cable from Washington to New Delhi ... be reported to me through the Toronto consulate. It arrived in code; no facilities existed for decoding. They brought it to me at the airport — a mass of numbers. I asked if they assumed I could read it. They said no. I asked how they managed. They said that when something arrived in code, they phoned Washington and had the original read to them.

## 14 Further reading

For many students this will be the last university mathematics course they will take. Although the twin subjects of error-correcting codes and cryptography occupy a small place in the grand panorama of modern mathematics, it seems to me that they form a very suitable topic for such a final course.

Outsiders often think of mathematicians as guardians of abstruse but settled knowledge. Even those who understand that there are still problems unsettled ask what mathematicians will do when they run out of problems. At a more subtle level Kline's magnificent *Mathematical Thought from Ancient to Modern Times* [4] is pervaded by the melancholy thought that though the problems will not run out they may become more and more baroque and inbred. 'You are not the mathematicians your parents were' whispers Kline 'and your problems are not the problems your parent's were.'

However, when we look at this course we see that the idea of error-correcting codes did not exist before 1940. The best designs of such codes depend on the kind of 'abstract algebra' that historians like Kline and Bell consider a dead end but lie behind the superior performance of CD players and similar artifacts.

In order to go further into both codes, whether secret or error correcting we need to go into the the question of how the information content of a message is to be measured. 'Information theory' has its roots in the code breaking of World War II (though technological needs would doubtless have lead to the same ideas shortly thereafter anyway). Its development required a level of sophistication in treating probability which was simply not available in the 19th century. (Even the Markov chain is essentially 20th century.)

The question of what makes a calculation difficult could not even have been thought about until Gödel's theorem (itself a product of the great 'foundations crisis' at the beginning of the 20th century). Developments by Turing and Church of Gödel's theorem gave us a theory of computational complexity which is still under development today. The question of whether there exist 'provably hard' public codes is intertwined with still unanswered questions in complexity theory. There are links with the profound (and very 20th century) question of what constitutes a random number.

Finally the invention of the electronic computer has produced a cultural change in the attitude of mathematicians towards algorithms. Before 1950, the construction of algorithms was a minor interest of a few mathematicians. (Gauss and Jacobi were consider unusual in the amount of thought they gave to actual computation.) Today we would consider a mathematician as much as a maker of algorithms as a prover of theorems. The notion of the *probabilistic algorithm* which hovered over much of our discussion of secret codes is a typical invention of the last decades of the 20th century.

Although both subjects are now 'mature' in the sense that they provide usable and well tested tools for practical application they still contain deep unanswered questions. For example

How close to the Shannon bound can a 'computationally easy' error correcting code get?

Do provably hard public codes exist?

Even if these questions are too hard there must surely exist error correcting and public codes based on new ideas. Such ideas would be most welcome and, although they are most likely to come from the professionals they might come from outside the usual charmed circles.

The best book I know for further reading is Welsh [9]. After this the book of Goldie and Pinch [7] provides a deeper idea of the meaning of information and its connection with the topic. The book by Koblitz [5] develops the number theoretic background. The economic and practical importance of transmitting, storing and processing data far outweighs the importance of hiding it. However, hiding data is more romantic. For budding cryptologists and cryptographers (as well as those who want a good read) Kahn's *The Codebreakers* [2] has the same role as is taken by Bell's *Men of Mathematics.* for budding mathematicians.

## References

[1] U. Eco *The Search for the Perfect Language* (English translation), Blackwell, Oxford 1995.

[2] D. Kahn *The Codebreakers: The Story of Secret Writing* MacMillan, New York, 1967. (A lightly revised edition has recently appeared.)

[3] D. Kahn *Seizing the Enigma* Houghton Mifflin, Boston, 1991.

[4] M. Kline *Mathematical Thought from Ancient to Modern Times* OUP, 1972.

[5] N. Koblitz *A Course in Number Theory and Cryptography* Springer, 1987.

[6] D. E. Knuth *The Art of Computing Programming* Addison-Wesley. The third edition of Volumes I to III is appearing during this year and the next (1998–9).

[7] G. M. Goldie and R. G. E. Pinch *Communication Theory* CUP, 1991.

[8] T. M. Thompson *From Error-correcting Codes through Sphere Packings to Simple Groups* Carus Mathematical Monographs **21**, MAA, Washington DC, 1983.

[9] D. Welsh *Codes and Cryptography* OUP, 1988.

## 15 First Sheet of Exercises

Because this is a third term course I have tried to keep the questions simple. On the whole Examples will have been looked at in the lectures and Exercises will not but the distinction is not very clear.

**Q 15.1.** Do Exercise 1.1. In the model of a communication channel we take the probability $p$ of error to be less than $1/2$. Why do we not consider the case $1 \geq p > 1/2$? What if $p = 1/2$?

**Q 15.2.** Do Exercise 2.3 Machines tend to communicate in binary strings so this course concentrates on *binary alphabets* with two symbols. There is no particular difficulty in extending our ideas to *alphabets* with $n$ symbols though, of course, some tricks will only work for particular values of $n$. If you look at the inner title page of almost any recent book you will find its International Standard Book Number (ISBN). The ISBN uses single digits selected from 0, 1, ... , 8, 9 and $X$ representing 10. Each ISBN consists of nine such digits $a_1$, $a_2$, ... , $a_9$ followed by a single check digit $a_{10}$ chosen so that

$$10a_1 + 9a_2 + \cdots + 2a_9 + a_{10} \equiv 0 \mod 11. \tag{*}$$

(In more sophisticated language our code $C$ consists of those elements $\mathbf{a} \in \mathbb{F}_{11}^{10}$ such that $\sum_{j=1}^{10}(11-j)a_j = 0$.)

(i) Find a couple of books and check that $(*)$ holds for their ISBNs.

(ii) Show that $(*)$ will not work if you make a mistake in writing down one digit of an ISBN.

(iii) Show that $(*)$ may fail to detect two errors.

(iv) Show that $(*)$ will not work if you interchange two adjacent digits.

Errors of type (ii) and (iv) are the most common in typing.

**Q 15.3.** Do Exercise 2.4 Suppose we use eight hole tape with the standard paper tape code and the probability that an error occurs at a particular place on the tape (i.e. a hole occurs where it should not or fails to occur where it should) is $10^{-4}$. A program requires about 10 000 lines of tape (each line containing eight places) using the paper tape code. Using the Poisson approximation, direct calculation (possible with a hand calculator but really no advance on the Poisson method) or otherwise show that the probability that the tape will be accepted as error free by the decoder is less than .04%.

Suppose now that we use the Hamming scheme (making no use of the last place in each line). Explain why the program requires about 17 500 lines of tape but that any particular line will be correctly decoded with probability about $1 - (21 \times 10^{-8})$ and the probability that the entire program will be correctly decoded is better than 99.6%.

**Q 15.4.** Show that if $0 < \delta < 1/2$ there exists an $A(\delta) > 0$ such that whenever $0 \leq r \leq n\delta$ we have

$$\sum_{j=0}^{r} \binom{n}{j} \leq A(\delta) \binom{n}{r}.$$

(We use weaker estimates in the course but this is the most illuminating.

**Q 15.5.** Show that the $n$-fold repetition code is perfect if and only if $n$ is odd.

**Q 15.6.** Let $C$ be the code consisting of the word 10111000100 and its cyclic shifts (that is 01011100010, 00101110001 and so on) together with the zero code word. Is $C$ linear? Show that $C$ has minimum distance 5.

**Q 15.7.** Write down the weight enumerators of the trivial code, the repetition code and the simple parity code.

**Q 15.8.** List the codewords of the Hamming (7,4) code and its dual. Write down the weight enumerators and verify that they satisfy the MacWilliams identity.

**Q 15.9.** (a) Show that if $C$ is linear then so are its extension $C^+$, truncation $C^-$ and puncturing $C'$ provided the symbol chosen to puncture by is 0.

(b) Show that extension and truncation do not change the size of a code. Show that it is possible to puncture a code without reducing the information rate.

(c) Show that the minimum distance of the parity extension $C^+$ is the least even integer $n$ with $n \geq d(C)$. Show that the minimum distance of $C^-$ is $d(C)$ or $d(C) - 1$. Show that puncturing does not change the minimum distance.

**Q 15.10.** If $C_1$ and $C_2$ are of appropriate type with generator matrices $G_1$ and $G_2$ write down a generator matrix for $C_1|C_2$.

**Q 15.11.** Show that the weight enumerator of $RM(d, 1)$ is

$$y^{2^d} + (2^d - 2)x^{2^{d-1}}y^{2^{d-1}} + x^{2^d}.$$

**Q 15.12.** Do Exercise 3.6 which shows that even if $2^n/V(n,e)$ is an integer, no perfect code may exist.

(i) Verify that

$$\frac{2^{90}}{V(90,2)} = 2^{78}.$$

(ii) Suppose that $C$ is a perfect 2 error correcting code of length 90 and size $2^{78}$. Explain why we may suppose without loss of generality that $\mathbf{0} \in C$.

(iii) Let $C$ be as in (ii) with $\mathbf{0} \in C$. Consider the set

$$X = \{\mathbf{x} \in \mathbb{F}_2^{90} : x_1 = 1,\ x_2 = 1,\ d(\mathbf{0}, \mathbf{x}) = 3\}.$$

Show that corresponding to each $\mathbf{x} \in X$ we can find a unique $\mathbf{c}(\mathbf{x}) \in C$ such that $d(\mathbf{c}(\mathbf{x}), \mathbf{x}) = 2$.

(iv) Continuing with the argument of (iii) show that

$$d(\mathbf{c}(\mathbf{x}), \mathbf{0}) = 5$$

and that $c_i(\mathbf{x}) = 1$ whenever $x_i = 1$. By looking at $d(\mathbf{c}(\mathbf{x}), \mathbf{c}(\mathbf{x}'))$ for $\mathbf{x}, \mathbf{x}' \in X$ and invoking the Dirichlet pigeon-hole principle, or otherwise, obtain a contradiction.

(v) Conclude that there is no perfect $[90, 2^{78}]$ code.

# 16 Second Sheet of Exercises

Because this is a third term course I have tried to keep the questions simple. On the whole Examples will have been looked at in the lectures and Exercises will not but the distinction is not very clear.

**Q 16.1.** An *erasure* is a digit which has been made unreadable in transmission. Why are they easier to deal with than errors? Find a necessary and sufficient condition on the parity check matrix of a linear $(n, k)$ code for it to be able to correct $t$ erasures and relate $t$ to $n$ and $k$ in a useful manner.

**Q 16.2.** Consider the collection $K$ of polynomials

$$a_0 + a_1\omega + a_2\omega^2$$

with $a_j \in \mathbf{F}_2$ manipulated subject to the usual rules of polynomial arithmetic and the further condition

$$1 + \omega + \omega^2 = 0.$$

Show by direct calculation that $F^* = F \setminus \{0\}$ is a cyclic group under multiplication and deduce that $K$ is a finite field.
[Of course, this follows directly from general theory but direct calculation is not uninstructive.]

**Q 16.3.** (i) Identify the cyclic codes of length $n$ corresponding to each of the polynomials 1, $X - 1$ and $X^{n-1} + X^{n-2} + \cdots + X + 1$.

(ii) Show that there are three cyclic codes of length 7 corresponding to irreducible polynomials of which two are versions of Hamming's original code. What are the other cyclic codes?

(iii) Identify the dual codes for each of the codes in (ii).

**Q 16.4.** Do Example 7.15.

(i) If $K$ is a field containing $\mathbb{F}_2$ then $(a + b)^2 = a^2 + b^2$ for all $a, b \in K$.

(ii) If $P \in \mathbb{F}_2[X]$ and $K$ is a field containing $\mathbb{F}_2$ then $P(a)^2 = P(a^2)$ for all $a \in K$.

(iii) Let $K$ be a field containing $\mathbb{F}_2$ in which $X^7 - 1$ factorises into linear factors. If $\beta$ is a root of $X^3 + X + 1$ in $K$ then $\beta$ is a primitive root of unity and $\beta^2$ is also a root of $X^3 + X + 1$.

(iv) We continue with the notation (iii). The BCH code with $\{\beta, \beta^2\}$ as defining set is Hamming's original (7,4) code.

**Q 16.5.** A binary non-linear feedback register of length 4 has defining relation

$$x_{n+1} = x_n x_{n-1} + x_{n-3}.$$

Show that the state space contains 4 cycles of lengths 1, 2, 4 and 9

**Q 16.6.** A binary LSFR of length 5 was used to generate the following stream

$$101011101100\dots$$

Recover the feedback polynomial by the Berlekamp-Massey method.

**Q 16.7.** Do Exercise 8.5 Consider the linear recurrence

$$x_n = a_0 x_{n-d} + a_1 x_{n-d-1} \cdots + a_{d-1} x_{n-1} \tag{*}$$

with $a_j \in \mathbb{F}_2$ and $a_0 \neq 0$.

(i) Suppose $K$ is a field containing $\mathbb{F}_2$ such that the auxiliary polynomial $C$ has a root $\alpha$ in $K$. Then $\alpha^n$ is a solution of $(*)$ in $K$.

(ii) Suppose $K$ is a field containing $\mathbb{F}_2$ such that the auxiliary polynomial $C$ has $d$ distinct roots $\alpha_1$, $\alpha_2$, ... , $\alpha_d$ in $K$. Then the general solution of $(*)$ in $K$ is

$$x_n = \sum_{j=1}^{d} b_j \alpha_j^n$$

for some $b_j \in K$. If $x_0, x_1, \dots, x_{d-1} \in \mathbb{F}_2$ then $x_n \in \mathbb{F}_2$ for all $n$.

(iii) Work out the first few lines of Pascal's triangle modulo 2. Show that the functions $f_j : \mathbb{Z} \to \mathbb{F}_2$

$$f_j(n) = \binom{n}{j}$$

are linearly independent in the sense that

$$\sum_{j=0}^{m} a_j f_j(n) = 0$$

for all $n$ implies $a_j = 0$ for $1 \leq j \leq m$.

(iv) Suppose $K$ is a field containing $\mathbb{F}_2$ such that the auxiliary polynomial $C$ factorises completely into linear factors. If the root $\alpha_u$ has multiplicity $m_u$ $[1 \leq u \leq q]$ then the general solution of $(*)$ in $K$ is

$$x_n = \sum_{u=1}^{q} \sum_{v=0}^{m(u)-1} b_{u,v} \binom{n}{v} \alpha_u^n$$

for some $b_{u,v} \in K$. If $x_0, x_1, \dots, x_{d-1} \in \mathbb{F}_2$ then $x_n \in \mathbb{F}_2$ for all $n$.

**Q 16.8.** Do Exercise 10.8 One of the most confidential German codes (called FISH by the British) involved a complex mechanism which the British found could be simulated by two loops of paper tape of length 1501 and 1497. If $k_n = x_n + y_n$ where $x_n$ is a stream of period 1501 and $y_n$ is stream of period 1497 what is the longest possible period of $k_n$? How many consecutive values of $k_n$ do you need to to specify the sequence completely?

**Q 16.9.** We work in $\mathbf{F}_2$. I have a secret sequence $k_1$, $k_2$, $\dots$ and a message $p_1$, $p_2$, $\dots$ , $p_N$. I transmit $p_1 + k_1$, $p_2 + k_2$, $\dots$ $p_N + k_N$ and then, by error, transmit $p_1 + k_2$, $p_2 + k_3$, $\dots$ $p_N + k_{N+1}$. Assuming that you know this and that my message makes sense how would you go about finding my message? Can you now decipher other messages sent using the same part of my secret sequence?

**Q 16.10.** Give an example of a homomorphism attack on an RSA code. Show in reasonable detail that the Elgamal signature scheme defeats it.

**Q 16.11.** I announce that I shall be using the Rabin-Williams scheme with modulus $N$. My agent in X'Dofdro sends me a message $m$ (with $1 \geq m \leq N - 1$) encoded in the requisite form. Unfortunately my cat eats the piece of paper on which the prime factors of $N$ are recorded so I am unable to decipher it. I therefore find a new pair of primes and announce that I shall be using the Rabin Williams scheme with modulus $N' > N$. My agent now recodes the message and sends it to me again.

The dreaded SNDO of X'Dofdro intercept both code messages. Show that they can find $m$. Can they decipher any other messages sent to me using only one of the coding schemes?

**Q 16.12.** Extend the Diffie-Helman key exchange system to cover three participants in a way that is likely to be as secure as the two party scheme.

Extend the system to $n$ parties in such a way that they can compute their common secret key in at most $n^2 - n$ communications.