MAT3343: Applied Algebra

An introduction to codes and cryptography

Monica Nevins

Contents

Acknowledgements

This volume grew from my lecture notes developed for MAT3343 and MAT3743 over the years. Each time I teach this course, I seem to cover a different set of topics; the subset here covers only error-correcting codes and cryptography, and is probably still more than will be covered in a single course!

My first lecture notes were heavily influenced by those of my colleague Peter Campbell, and on the treatment of groups and codes in the lovely book by Keith Nicholson [Nic12]. Further iterations have also been enriched by the lecture notes for this course at the University of Ottawa by my colleague Michael Newman, and by the lecture notes for a related course at Carleton University taught by Daniel Panario. It is a pleasure to thank these people.

Your corrections to the text are quite welcome!

Monica Nevins
mnevins@uottawa.ca

Preface

Applications of algebra abound! There is a reason that much of 20th century mathematics was devoted to the development of such fruitful areas as algebraic geometry, algebraic topology, algebraic combinatorics, algebraic number theory... really, algebraic anything.

In this text, we focus on applications relating to error-correcting codes, and to cryptography. Our treatment is meant to give a taste of each of these subjects, with enough depth to appreciate the power of the algebraic tools we bring to bear on the problems in the field. While these applications are the focus, an equally important goal is to help the reader deepen their knowledge of algebra by seeing how it can be used in unexpected ways.

Each chapter includes one or more sections labeled “Exercises". As every math student knows, it is by solving problems that one really learns the material. The exercises generally reflect the preceding few sections in order, and generally increase in level of difficulty — but this is not a rule! Do browse them all.

The text is organized as follows. In Part I, after a brief recap of some key tools from group theory and linear algebra, we define linear error-correcting codes, and establish the key notions of the field, including Hamming distance, error-correction, generator matrices, parity check matrices and syndromes.

With some appreciation that it is difficult to construct good codes, we then turn to cyclic codes in Part II. Here, we need to develop the theory of polynomial rings over a finite field, including ideals and quotient rings, in order to classify all cyclic codes. We then dive further into the theory of quotient rings and construct finite fields beyond $\mathbb{Z}/p\mathbb{Z}$ (for p a prime). The exploration of their structure leads to the celebrated BCH theorem and the construction of BCH and Reed–Solomon codes: an infinite class of codes that we can construct to suit our specifications for length and error-correction capability.

Finally, in Part III we switch gears to discuss another problem in communication theory, namely that of ensuring the privacy of communication using cryptography. We quickly focus on public-key encryption and set it in the context of post-quantum cryptography. We present three major “post-quantum" cryptographic systems: NTRU, Classic McEliece and Kyber (based on LWE), all of which relate to the themes and tools of the first two parts.

The appendices include some samples of mathematical background expected from prerequisite courses, and an additional section on elliptic curves over finite fields (with a focus on Elliptic Curve Cryptography). Although Appendix C purports to have solutions, only a handful have been written.

The text concludes with a bibliography of all references cited, for your independent reading and research, as well as a detailed index to allow you to quickly find the definitions of key terms.

We hope you are inspired by what you find here, and explore further!

Monica Nevins
University of Ottawa
April 2023

Codes

1. Introduction

The word “code” evokes two essentially opposite ideas:

- increasing the clarity of a message (so that the recipient can decode it, even when the message is damaged in transmission)
- obfuscating a message (so that none but the intended recipient can decode it)

Both are key important problems in communications theory, and we will study both in this course. The first is the domain of *error-correcting codes*; the second is *cryptography*. A reference for the material on codes is [Nic12, Chapter 2.11] and [Nic12, Chapter 6.4].

So let us start with codes. The setting is as follows.

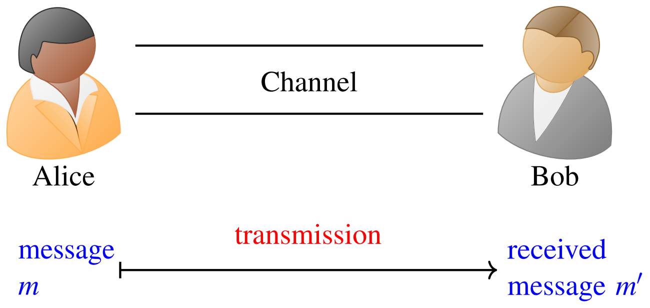


Figure 1.1: A sender transmitting message m to a receiver.

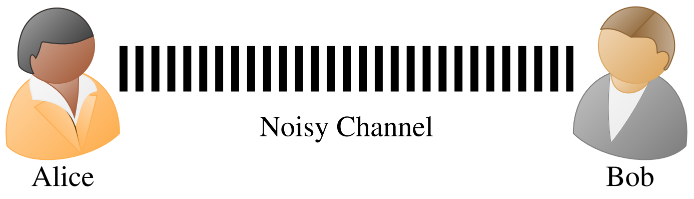
A sender (Alice) has a message m which is to be conveyed to a recipient (Bob). The process of conveying the message, which we’ll call *transmission* could take the form of, for example,

- voice in the air,
- written text on paper,
- keystrokes typed in a computer,
- electrical impulses along a wire, or

- modulated radio waves through the air

among hundreds of other possibilities. The means of conveyance (such as air, paper, keyboard, or wire) is called the *channel*. This would be represented in Figure 1.1.

But suppose the channel is *noisy*, meaning that it can distort the transmission; can we nonetheless ensure accurate communication? That is, we may add an additional step, as in Figure 1.2.



$$\begin{tikzcd} \text{message } m \arrow[d, mapsto, "\text{encoding}"] & & \text{received message } m' \\ \text{codeword } c \arrow[rr, mapsto, "\text{transmission}"] & & \text{received word } c' \arrow[u, mapsto, "\text{decoding}"'] \end{tikzcd}$$

Figure 1.2: The message m is first encoded to the codeword c, which is transmitted. What is received is c', which is $c+n$, where n is some noise. Decoding c' yields a message m'. The communication is successful if $m = m'$.

So the sender has a message m. We could think of it as a block of text, or a computer file. The sender *encodes* the message to produce the encoded message c, which is a *codeword* (or may be several codewords, if we have broken our long message into several *blocks*). The codeword is transmitted through the channel, and is picked up by the receiver. What the receiver collects is called the *received word* c'; if the channel was noisy then it is possible that the received words are not identical to the sent codewords. The receiver then *decodes* the received word (which involves, roughly, guessing which codeword was sent) and then translates the codeword back to the corresponding message m'.

The problem of error-correcting codes is to provide a *code*, which is a set of codewords, and a *decoding algorithm*, which gives a maximal likelihood of success of accurate transmission of a message (that is, that $m = m'$), under given circumstances. We are thus most interested in codes which can detect if there was an error in transmission (*error-detecting codes*) or, even better, can correct an error in transmission (*error-correcting codes*).

So for example, if the communication is voice in the air: noise could correspond to noise in the environment. Possible codes include: repeating long words, or using longer sentences, or switching to a language that stands out better against the background[1]

[1] Speaking more loudly is equivalent to amplifying the signal to drown out the noise; technically possible in all systems but usually at some cost (in this case, dignity). Using sign language or gestures is another possible solution; this corresponds to switching to a different, less noisy, channel.

If it's typing on a computer keyboard, maybe the noise issue is about the possibility of mistyping numbers (transposition errors); if it's transmission on a wire or on radio waves, it can be about interference from other sources. How does this interference cause errors in a digital message? In fact, if the message is a sequence of 0s and 1s, then this could be converted into a square wave pattern with a well-defined frequency, as in Figure 1.3. When this is transmitted, the square wave loses power over

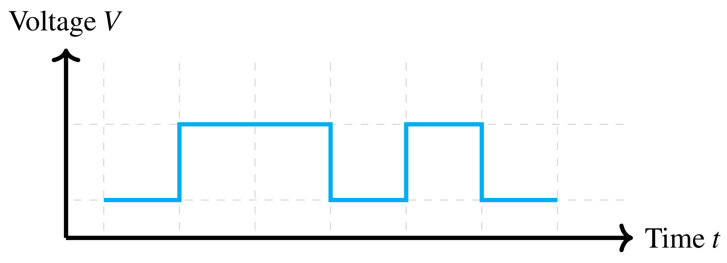


Figure 1.3: Digital data transmitted as a square wave (at a given frequency, with periods indicated by vertical dashed lines). Each high represents a 1, each low represents a 0. This message is the byte 011010.

long distances and noise at the modulation frequency will be added to the signal. The result is that the received wave is no longer square; the receiver uses thresholds to guess if the signal is high or low on each time period. Without the use of error-correcting codes, digital transmission would have no

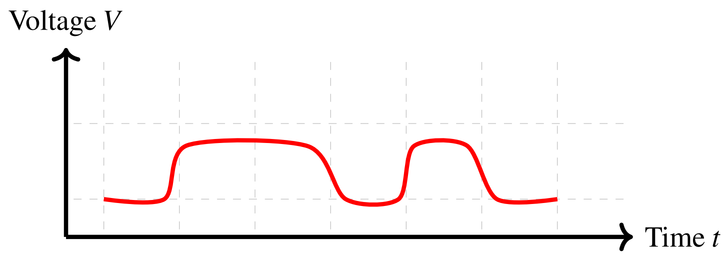


Figure 1.4: Transmission degrades digital signals.

advantage over analog transmission (where you just accept the noise into your received signal: hiss, crackle, pop). This is the case for AM/FM radio for example.

For the first part of the course, we will make the assumption that the errors are randomly distributed and occur with small probability; for example, we might expect at most one error per codeword. Much later, we will discuss codes that are more suitable under conditions of "bursts" of errors (where several consecutive bits may be destroyed, for example).

■ **Example 1.1** ASCII is a code for converting messages which are letters or characters into binary numbers, which can then be transmitted. For example, the messge 'a' corresponds to the codeword 1100001. However, if one bit is damaged, for example:

- 1110001 = 'q'
- 1100101 = 'e'
- 0100001 = '1'

then we wouldn't know, except from the context of the message, that an error had occured[2]. So ASCII is not error-detecting. ■

■ **Example 1.2** Morse code is a code for converting messages which are letters into sequences of dots and dashes, which can be transmitted by light or electrical signals. For example, the message 'ℓ' is the codeword . – .. (dot-dash-dot-dot). If an error occurs in the third symbol, you get . – –., which corresponds to 'p'. But if the error occurs in the last symbol, you get . – .– which is not a codeword (that is, it does not correspond to any letter). So Morse code exhibits *some* error detection. ■

■ **Example 1.3** Suppose the message is already a binary number. We could encode it by repeating it twice: the message 011 becomes the codeword 011011. Then if one bit is damaged in transmission, the receiver will immediately detect an error, because the first three bits would not be identical to the second three bits. So the receiver can detect all *single* errors (but can only detect *some* double errors). However, the receiver can't correct the error; it has no way to deciding which of the two halves of the message are correct. ■

■ **Example 1.4** In the same setting as above: suppose the message is a single bit 0 or 1, and the corresponding codeword is obtained by repeating it THREE times. So 0 is encoded as 000 and 1 as 111. Then the receiver can detect *and correct* a single error, by using a majority vote on which bit the message should be. ■

[2]And now imagine that the message itself was encrypted — then you wouldn't even have the context to tell you there was an error. This is why we emphasize the case where the receiver can work out the error detection/correction on a word-by-word basis.

2. Algebraic structures

Our codes will have algebraic structure — these structures are what will provide patterns, regularity, designability, efficiency, and much more. In this chapter, we give a brief introduction to some familiar, and less familiar, algebraic structures that we'll need.

2.1 Groups

The fundamental algebraic structure is that of a group, which is a set with one operation, like addition, or multiplication. Since the notation looks quite different in these two cases, we define them separately, but note that the axioms are the same!

Definition 2.1 An *additive group* is a set G equipped with an operation denoted $+$ satisfying the following axioms:
closure: $\forall a,b \in G$, $a+b \in G$;
commutativity: $\forall a,b \in G$, $a+b=b+a$;
associativity: $\forall a,b,c \in G$, $(a+b)+c = a+(b+c)$;
identity: $\exists z \in G$ such that $\forall a \in G$, $a+z=a$ — so we write $z=0$ for this element;
inverse: $\forall a \in G, \exists b \in G$ such that $a+b=0$ — so we write $b=-a$ for this element.

■ **Example 2.2** We know many additive groups:

- $\mathbb{Z}$, the integers;
- $\mathbb{Z}/n\mathbb{Z}$, the integers mod n, for any $n>1$;
- any vector space V;
- the set of $m \times n$ matrices, for any $m,n \geq 1$.

■

■ **Example 2.3** In contrast, $\mathbb{N}$, the set of natural numbers is not a group, because we are missing the additive inverses. ■

We can also write groups with multiplicative notation. In this case, by convention, we do NOT automatically assume that the multiplication is commutative.

Definition 2.4 A *group* is a set G equipped with an operation (denoted $\cdot$ or by juxtaposition) satisfying the following axioms:
closure: $\forall a, b \in G$, $ab \in G$;
associativity: $\forall a, b, c \in G$, $(ab)c = a(bc)$;
identity: $\exists u \in G$ such that $\forall a \in G$, $ua = a = au$ — so we denote this element by 1 or e;
inverse: $\forall a \in G, \exists b \in G$ such that $ab = 1 = ba$ — so we denote this inverse by $b = a^{-1}$.
If in addition the operation satisfies
commutativity: $\forall a, b \in G$, $ab = ba$
then it is called an *abelian group*.

■ **Example 2.5** The set $\mathbb{R}^\times$ of nonzero real numbers is an abelian group. The set of invertible 2×2 matrices over $\mathbb{R}$ is a nonabelian group. ■

A *subgroup* of G is a subset H that is also a group under the same operation. It suffices to verify that H is closed under the operation and under taking inverses. (Exercise)

2.2 Finite Fields

What about two operations? It will turn out (see Chapter 6) that there is a huge variety of interesting axioms we could ask about a set equipped with two operations (traditionally denoted $+$ and $\cdot$), but the simplest is that of a field.

Definition 2.6 A *field* is a set F equipped with two operations, addition and multiplication, such that

1. F is an additive group, with identity element 0;
2. $F^\times := F \setminus \{0\}$ is an abelian group under multiplication; and
3. multiplication distributes over addition, that is, $\forall a, b, c \in F$, $a(b+c) = ab + ac$.

■ **Example 2.7** The set of real numbers $\mathbb{R}$ is a field, as are the complex numbers $\mathbb{C}$ and the rational numbers $\mathbb{Q}$. ■

■ **Example 2.8** The set of integers $\mathbb{Z}$ is not a field, since 2 has no multiplicative inverse in $\mathbb{Z}$, so $\mathbb{Z} \setminus \{0\}$ is not a group under multiplication. ■

■ **Example 2.9** The set of 2×2 matrices is not a field, since there are nonzero elements which are not invertible (and since multiplication is not commutative!)[1]. ■

■ **Example 2.10** The set $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$, with componentwise addition and multiplication, is not a field, because there exist non-zero elements, like $(0,1)$, which are not invertible. ■

The most important examples of fields for coding theory are the finite ones.

■ **Example 2.11** The set $\mathbb{Z}_2 = \{0, 1\}$ with addition and multiplication mod 2, is a field, called the *binary field*. It is in fact the smallest possible field (since every field contains 0 and 1 and we have

[1]This latter transgression can be forgiven if all other properties hold, in which case we insist on distributivity in both directions and call it a *skew field*. The quaternions are an example.

$0 \neq 1$). ■

Theorem 2.12 Let p be a prime number. Then the set $\mathbb{Z}_p$, with addition and multiplication mod p, is a field.

Proof. We know that $\mathbb{Z}_p$ is an additive group, with identity 0. Distributivity of multiplication over addition follows from that in $\mathbb{Z}$, since $a(b+c) = ab+ac$ in $\mathbb{Z}$ implies $a(b+c) \equiv ab+ac \mod p$.

Now set $\mathbb{Z}_p^\times = \{1, 2, \cdots, p-1\}$. We first need to show that multiplication is well-defined on $\mathbb{Z}_p^\times$. Clearly the product mod p gives a result in $\mathbb{Z}_p$; we need to prove that if neither a nor b is zero then $ab \neq 0$ as well. Let $a, b \in \mathbb{Z}_p^\times$. We have

$$\begin{aligned} ab \equiv 0 \mod p &\Leftrightarrow p \text{ divides } ab \\ &\Leftrightarrow p \text{ divides } a \text{ OR } p \text{ divides } b \\ &\Leftrightarrow a \equiv 0 \mod p \text{ OR } b \equiv 0 \mod p. \end{aligned}$$

Note that we used the theorem on uniqueness of prime factorization to prove this; you can see where it would fail if p were not prime. Therefore $ab \in \mathbb{Z}_p^\times$, and $\mathbb{Z}_p^\times$ is closed under multiplication mod p.

The unit is 1, which lies in $\mathbb{Z}_p^\times$; and multiplication mod p in $\mathbb{Z}_p$ is associative and commutative since multiplication is associative and commutative in $\mathbb{Z}$.

For inverses: recall that the extended Euclidean algorithm says that if $\gcd(a,b) = d$ then there exist integers x, y such that

$$ax + by = d. \tag{2.1}$$

Here, set $b = p$ and $a \in \mathbb{Z}_p^\times$; then $d = 1$. Hence we have x, y so that

$$ax + py = 1;$$

taking this equation mod p yields $ax \equiv 1 \mod p$, or equivalently, $x \equiv a^{-1} \mod p$. We conclude that every element of $\mathbb{Z}_p^\times$ has an inverse, so this is a group; it follows that $\mathbb{Z}_p$ is a field. ■

Corollary 2.13 If $n > 1$ is not prime, then $\mathbb{Z}_n$ is not a field.

Proof. If n is not prime, then $n = ab$ for some $1 < a, b < n$. Thus n does not divide either a or b, so neither is 0 mod n, but their product is. Thus $\mathbb{Z}_n \setminus \{0\}$ is not closed under multiplication. ■

R A natural question is: what other finite fields are there? We'll answer this in Chapter 8, when we want to design codes based on finite fields beyond $\mathbb{Z}_2$ and $\mathbb{Z}_p$.

2.3 Vector spaces

Now how about THREE operations? That's asking a lot; but vector spaces come close.

Definition 2.14 Let F be a field. A *vector space* over F is a set V equipped with *scalar multiplication* by elements of F and vector addition such that

1. V is an additive group;
2. $\forall a, b \in F, \forall v, w \in V$, we have $av \in V$ and
   - $(a+b)v = av + bv$;
   - $(ab)v = a(bv)$;
   - $a(v+w) = av + aw$;
3. $\forall v \in V$, $1v = v$.

■ **Example 2.15** The set of column vectors of length n, $\mathbb{R}^n$, is a vector space over $\mathbb{R}$. The set of polynomials of degree n, $\mathscr{P}_n = \{a_0 + a_1x + a_2x^2 + \ldots + a_nx^n | a_i \in \mathbb{R}\}$, is a vector space over $\mathbb{R}$. ■

■ **Example 2.16** In the same vein: for any field F (such as $F = \mathbb{Z}_2$), the set $F^n = \{(a_1, a_2, \cdots, a_n) \mid a_i \in F\}$ is a vector space over F, as are the set of polynomials $\mathscr{P}_n = \{a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \mid a_i \in F\}$. ■

Recall that a *subspace* is a subset of a vector space that is a vector space in its own right. The *subspace test* is: if a subset W of a vector space V satisfies the three conditions:

- W is closed under addition;
- W is closed under scalar multiplication;
- W is nonempty;

then W is a subspace.

■ **Example 2.17** Let $W = \{(x,y,z) \in \mathbb{Z}_7^3 \mid x + y + z = 0\}$. By the subspace test (Exercise), this is a subspace of $\mathbb{Z}_7^3$, hence a vector space over $\mathbb{Z}_7$. ■

We will use many familiar facts about vector spaces in this course — but always applied to vector spaces over finite fields. Some examples (stated for a vector space V over a field F) include:

Linear independence: a set $\{v_1, \cdots, v_n\}$ of vectors in V is *linearly independent* if and only if the only solution to the equation $a_1v_1 + \cdots + a_nv_n = 0$, with $a_i \in F$, is the trivial solution $a_1 = a_2 = \cdots = a_n = 0$.

Span: a set $\{v_1, \cdots, v_n\}$ of vectors in V *spans* V if and only if every element $w \in V$ can be written as $w = \sum_i a_iv_i$ in at least one way, with $a_i \in F$.

Basis: a basis for V is a maximal linearly independent set of vectors; or a minimal spanning set of vectors; or any spanning set that is linearly independent.

Dimension: Every vector space has a basis. When the basis is finite, its cardinality is the *dimension* of V and is independent of the choice of basis. (We will only consider finite dimensional vector spaces in this course.)

Matrices: A linear map $T\colon V \to W$ between vector spaces may be represented by a matrix M, as follows. Choose a basis $\{e_1, \cdots, e_n\}$ of V and a basis $B = \{f_1, \cdots, f_m\}$ of W. Then M is the $m \times n$ matrix whose ith column is the coordinate vector of $T(e_i)$ with respect to B.

These properties are proven in any introductory linear algebra text, such as [GJN21], for vector spaces over the real numbers. The proofs go through unchanged for vector spaces over other fields.

Notation for vector spaces If p is a prime then we saw $\mathbb{Z}_p^n$ is an n-dimensional vector space over $\mathbb{Z}_p$. Let's write

$$v_1 v_2 \cdots v_n \quad \text{instead of} \quad (v_1, v_2, \cdots, v_n)$$

for elements of $\mathbb{Z}_p^n$, when convenient. So for example, 0011 represents the vector $(0,0,1,1)$ in $\mathbb{Z}_2^4$.

In particular, note that

$$0011 + 0101 = 0110 \qquad \text{NOT} \qquad 1000.$$

That is to say: remember that when we add vectors, the addition is XOR (componentwise), not the addition of numbers in base 2 with carrying.

2.4 Exercises

1. Choose a set of representatives for $\mathbb{Z}_5$ and write down its addition and multiplication tables.
2. Show that $\mathbb{Z}_6$ is an additive group. Show that $\mathbb{Z}_6$ is not a group under multiplication, by identifying all elements that do not have a multiplicative inverse mod 6.
3. Show explicitly that $\mathbb{Z}_3$, with addition and multiplication mod 3 is a field, but that $\mathbb{Z}_4$ is not.
4. Suppose we define $F = \{0, 1, x, 1+x\}$ with addition of polynomials with coefficients in $\mathbb{Z}_2$, and multiplication as for polynomials except we declare that $x^2 = 1 + x$, so the answer is again in F. Prove this is a field with 4 elements. Prove that $F \ncong \mathbb{Z}_4$.
5. Prove that in a field F, we have for all $a \in F$ that $a(-1) = -a$. Here, -1 is the additive inverse of 1, and $-a$ is the additive inverse of a.
6. Prove that if $n = ab$ is a proper factorization of n, then neither a nor b has a multiplicative inverse mod n. Hint: try an argument by contradiction.
7. Prove that the usual 10 axioms which define a vector space over a field are equivalent to the shortened version given in Definition 2.14.
8. Find a basis for $W = \{(x,y,z) \in \mathbb{Z}_7^3 \mid x + y + z = 0\}$. What is $\dim(W)$?
9. How many vectors are there in a vector space of dimension n over a finite field with q elements?
10. Suppose $F \subset E$ are fields with the same operations. Prove that E is a vector space over F. This has profound and very practical implications!

3. Linear codes

In our discussion of coding, we saw that if we wish to have error-detection in our codes, then our valid codewords should only be a subset of all possible received words (with the hopes that an error is likely to turn a valid codeword into something invalid that we can therefore detect).

3.1 Definitions and examples of linear codes

So a code is a set of points in F^n, where F is some alphabet of symbols. Let us give this more structure, and make this more formal and more algebraic, with the following definition.

Definition 3.1 A *linear block code* (or just: *linear code*) over the finite field F is a **subspace** C of F^n. Each element $c \in C$ is called a *codeword*. If $\dim(C) = k$, where $k \leq n$, then it is called an (n,k)-code over F. If $|F| = q$ then C is called a q-ary linear code. The *length* of the code is n. The *size* of C is the number of codewords in C. Its *code rate* is $R = k/n$, a measure of the efficiency of encoding.

R We say “binary” in place of 2-ary and “ternary” in place of 3-ary.

In this model, the process of encoding is the process of assigning a codeword for each message. We would commonly think of breaking our message into smaller blocks, and encoding each block one at a time, and transmitting it as soon as we’ve encoded it[1].

■ **Example 3.2** ASCII consists of all 7-bit codewords in $\mathbb{Z}_2^7$. So this is a $(7,7)$ binary code. It is full rate. It cannot detect any errors. ■

[1] Note that block codes are only one possible model for communications; other coding schemes may require the entire message to be encoded before any part is transmitted; or may use the previously transmitted blocks to help encode the current block. That said, the block model is the fastest and most popular method for coding in use.

■ **Example 3.3** ASCII with parity check is the set

$$C = \{v \in \mathbb{Z}_2^8 \mid \sum_{i=1}^{7} v_i = v_8\}$$

(where of course the sum is mod 2). Since the equation defining elements of C is linear, we deduce that C is a subspace of $\mathbb{Z}_2^8$; since there is only one (nontrivial) linear equation, we deduce that $\dim(C) = 7$. Thus this is an $(8,7)$ binary code and its rate is $7/8$.

A message is an ASCII character, which is encoded by appending its parity check bit. For example, to encode the message $m =' e'$, we first look up its ASCII code, which is 1100101. These are $v_1 v_2 \cdots v_7$. Their sum is $4 = 0 \bmod 2$; thus the parity check bit is 0 and our codeword is 11001010.

This code is single error detecting because if exactly one bit is changed (from a 0 to a 1 or vice versa) then it will no longer be true that the sum of the digits is 0 mod 2, so the result will not be a codeword. Ergo: an error occurred. ■

R The parity check code could also detect 3 errors, or any odd number of errors, but since it can't detect 2 errors that's a strict upper bound and we say it's only single-error-detecting.

■ **Example 3.4** The repetition code $C = \{000, 111\}$ is a one-dimensional subspace of $\mathbb{Z}_2^3$, so is a $(3,1)$ binary code. We saw that this code is single error correcting. It has rate $1/3$. ■

■ **Example 3.5** ISBN code (International Standard Book Number): this is a 10-digit number (although some are longer) where:

- the first digit represents the country;
- the next three represent the publisher;
- the next 6 represent the book;
- the last is a check digit.

ISBN 0-8048-1905-X

51295

9 780804 819053

Figure 3.1: An example of a 10-digit ISBN code.

Here, C is defined as the subspace of $\mathbb{Z}_{11}^{10}$ consisting of all vectors $a = (a_1, \cdots, a_{10})$ which satisfy

$$\sum_{i=1}^{10} i a_i \equiv 0 \mod 11.$$

This is a $(10,9)$ linear 11-ary code. When we write an element of $\mathbb{Z}_{11}$, we use X in place of 10, so that $\mathbb{Z}_{11} = \{0, 1, \cdots, 9, X\}$. In practice, we don't use X except as a last digit.

Let's check that $080481905X \in C$. We compute

$$\begin{aligned}
1\times 0+2\times 8+3\times 0+4\times 4+5\times 8+6\times 1+7\times 9+8\times 0+9\times 5+10\times X \\
&= 0+16+0+16+40+6+63+0+45+100 \\
&= 0+5+0+5+7+6+(-3)+0+1+1 \quad \text{simplifying mod } 11 \\
&= 22 \\
&\equiv 0 \mod 11
\end{aligned}$$

as required.

We can see from examples that this code detects one error, just as the parity check code does (and we can prove this algebraically). But it does a little more than the parity check code does: it can detect transposition errors. That is, if two digits are swapped in the course of entry, then their weighted sum will no longer be 0. This particularly kind of error detection is an important one in this context!

That said, it cannot correct errors: you can create two different ISBNs that give the same result after a single error (in different locations). ■

■ **Example 3.6** Consider the following subset of $\mathbb{Z}_2^6$:

$$C = \{000\,000,\ 001\,110,\ 010\,101,\ 100\,011,\ 011\,011,\ 101\,101,\ 110\,110,\ 111\,000\}.$$

We claim this is a linear code. Since the field is just $\mathbb{Z}_2$, it would suffice to check that this set is closed under addition, since vector spaces over $\mathbb{Z}_2$ are just additive groups (that is, the action by scalars is trivial). But it's handier to think of vector spaces.

Namely, every k-dimensional vector space over $\mathbb{Z}_2$ has 2^k elements. Thus, if C is a vector space, we must have $k = 3$. This tells us any basis must have 3 vectors, so we need to find three linearly independent vectors.

The first three vectors (after the zero vector) are clearly linearly independent (look at the first three bits). To prove they form a basis, it suffices to verify that the remaining 4 vectors are the various sums of these basis vectors, which they are.

Wasn't that a lot easier than applying the subspace test?

This is a $(6,3)$ binary linear code. We'll explore it in more detail in a bit. ■

This formalizes the notion of a code, and encoding a message. But what are we looking for in codes? What about the channel and decoding?

3.2 Binary symmetric channels

We assume that we are using a *binary symmetric channel*, which assumes that we communicate digitally and errors occur randomly and independently on each bit, with a small probability p. In particular, if our code is not a binary code, then there is another step (which we treat as a black box and will always ignore) that converts our codeword to a binary sequence, and then converts the received vector from binary to our alphabet.

Claude Shannon[2] defined the field of *information theory*, and established the main theorems of it, in a paper published in 1948 [Sha48]. In this paper, he precisely quantified the notion of "how much information is contained in a message" as the (probabilistic) *entropy* of the corresponding random variable X. This is something like the uncertainty on X. When you condition this variable on the random variable of what you receive, Y, your entropy decreases and the difference is called the *mutual information* — it tells you how much about the input is conveyed by the output. The *channel capacity* is the maximum over X,Y of the mutual information; it's a value between 0 and 1.

For our binary symmetric channel with error rate p, with X uniformly distributed across all bit strings, the entropy on X is defined as

$$H_2(p) = -(p\log_2(p) + (1-p)\log_2(1-p)). \tag{3.1}$$

and in the end the channel capacity comes down to

$$\mathscr{C} = 1 - H_2(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p).$$

(For more details, see [Mac03].)

Finally: we have seen that a good decoding algorithm has some built-in redundancy that can mean that you recover your original message even in the presence of noise on the channel. We call it a *block error* if the decoding algorithm does not recover the original message (on a given block).

The major theorem, first proven by Shannon, that ties this to codes is the following [Mac03, 10.1].

Theorem 3.7 — Noisy channel coding theorem. Given a channel with capacity $0 < \mathscr{C} \leq 1$, a value $\varepsilon > 0$, and a target rate of $R < \mathscr{C}$, then for large enough n, there exists a code of length n and rate at least R, and a decoding algorithm, such that the probability of block error is less than ε.

That's a lot to unpack: but basically: test your channel and estimate p, the probability of bit error; this lets you calculate the entropy $H_2(p)$ and thus the channel capacity $\mathscr{C}$. Choose a target code rate $R < \mathscr{C}$ that is acceptable to you. If you want 99.999% reliability, then you set $\varepsilon = 0.00001$. The theorem says: ok, no problem, there does exist an error-correcting code that will meet these specifications!

That's a fantastic result! Where's the loophole? Well, the code exists but we don't know how to construct it. Moreover, the code length n may be extraordinarily large. Ergo: there is a lot of value in analysing codes, and there are some pretty amazing codes out there, somewhere.

3.3 Maximum-likelihood decoding

So on a binary symmetric channel, in an n-bit codeword, a 1-bit error occurs with probability

$$P(1 \text{ error}) = p(1-p)^{n-1}$$

but any 2-bit error occurs with probability

$$P(2 \text{ errors}) = p^2(1-p)^{n-2} = \left(\frac{p}{1-p}\right) P(1 \text{ error}) \ll P(1 \text{ error}),$$

[2] https://en.wikipedia.org/wiki/Claude_Shannon

and so on. Thus we use a *maximum-likelihood decoder (ML)*, which means: if we receive a vector w then we should decode it to the codeword in C which has the most bits in common with w.

In a linear code, we can make this more precise. We model the introduction of errors by adding an *error vector* to the codeword; that is, the codeword c was transmitted, but the vector $x = c + z$, for some error vector z, is received at the receiver. Then ML decoding can be formalized as identifying the “closest" codeword to x, or “shortest" possible error vector z ... and to do this, we need a new notion of distance for this context.

3.4 Hamming weight and Hamming distance

Definition 3.8 The *Hamming weight* of a vector $x \in F^n$ is defined by

$$wt(x) = \#\{i \mid x_i \neq 0\},$$

the number of nonzero components of x.

■ **Example 3.9** In $\mathbb{Z}_2^8$, we have $wt(00100111) = 4$.

In $\mathbb{Z}_5^6$ we have $wt(310124) = 5$. ■

Note that two vectors have the same Hamming weight if and only if they have the same number of nonzero components; the actual values of these nonzero components is irrelevant.

The Hamming weight is the analogue of the *norm* of a vector, which measures its size. If we can measure size, then we should be able to measure distance, as well!

Definition 3.10 The *Hamming distance* between vectors $x, y \in F^n$ is defined by

$$d(x,y) = wt(x-y) = \#\{i \mid x_i \neq y_i\},$$

that is, the number of components in which x and y differ.

■ **Example 3.11** In $\mathbb{Z}_3^3$ we have $d(121,101) = 1$ and $d(012,120) = 3$. ■

This is an unusual notion of distance, as compared with the usual Euclidean distance on $\mathbb{R}^n$. However, it is essentially the only meaningful one! (See the exercises.)

What is amazing is that this primitive notion of distance is actually a metric.

Lemma 3.12 The Hamming distance is a metric on F^n. Moreover, it satisfies

$$\forall x \in F^n, wt(x) = d(x,0).$$

Recall: a *metric* d on a set M is a function $d\colon M \times M \to \mathbb{R}$ which satisfies, for all $x, y \in M$, that

(i) $d(x,y) \geq 0$;
(ii) $d(x,y) = 0$ iff $x = y$;
(iii) $d(x,y) = d(y,x)$; and

(iv) for all $z \in M$, $d(x,y) \le d(x,z) + d(z,y)$ (the triangle inequality).

Proof. The first three properties of a metric are immediate from the definition. To show the last one, suppose $x,y,z \in F^n$, and that $d(x,y) = d$. That is, x and y differ in exactly d components; say $T_{x,y} = \{i \mid x_i \neq y_i\}$.

Suppose $d(x,z) = d_1$ and $d(z,y) = d_2$. Let $T_x = \{i \mid x_i \neq z_i\}$ and let $T_y = \{j \mid y_j \neq z_j\}$. Set $T = T_x \cup T_y$, this has at most $d_1 + d_2$ elements (but could have fewer, if the sets intersect).

If $k \notin T$, then $x_k = z_k$ and $z_k = y_k$, so $x_k = y_k$. Therefore $k \notin T_{x,y}$. The contrapositve says: $T_{x,y} \subset T$. So $|T_{x,y}| \le |T|$, which implies $d \le |T| \le d_1 + d_2$, as required.

Finally: $d(x,0) = wt(x-0) = wt(x)$; this is just a restatement of the definition. ■

Once we have a metric, we can apply our favourite tools from topology.

Definition 3.13 Let $x \in F^n$. The *open ball of radius r centered at x* is

$$B_r(x) = \{y \in F^n \mid d(x,y) < r\}.$$

The *closed ball of radius r centered at x* is

$$S_r(x) = \{y \in F^n \mid d(x,y) \le r\}.$$

Note that if $F = \mathbb{R}$ and $d(x,y) = \|x-y\|$, then this is the usual notion of an open ball or closed ball. With a discretely valued metric such as the Hamming distance, however, the distinction between "open" and "closed" vanishes, and the radius is a bit flexible. They also don't really look like "balls" with a "centre". Basically, no part of the phrase "open ball of radius r centered at x" retains its meaning from $\mathbb{R}$!

■ **Example 3.14** Over $\mathbb{Z}_2$:

$$B_2(000) = \{000, 001, 010, 100\} = B_{1.5}(000) = S_1(000) = S_{1.5}(000).$$

So this set of 4 points is an open ball of radius 2, or 1.5; it is also a closed ball of radius 1, or 1.5. We draw this in Figure 3.2. ■

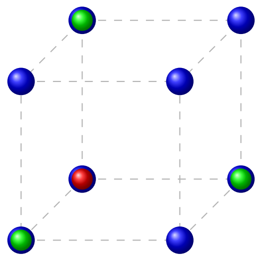


Figure 3.2: The closed ball of radius 1 centered at $(0,0,0)$ (in red) in $\mathbb{Z}_2$ is indicated by the red and green dots. Each green dot is a distance of exactly 1 from the red dot; each blue dot is at a distance of ≥ 2 from the red dot.

■ **Example 3.15** Over $\mathbb{Z}_3$, as depicted in Figure 3.3:

$$B_2(000) = \{000, 001, 010, 100, 002, 020, 200\} = S_1(000).$$

■

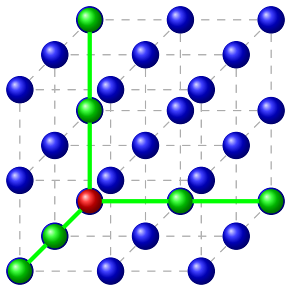


Figure 3.3: The closed ball of radius 1 centered at $(0,0,0)$ (in red) in $\mathbb{Z}_3$ is indicated by the red and green dots (connected by a green line as a visual aid). Each green dot is a Hamming distance of exactly 1 from the red dot; each blue dot is at a Hamming distance of ≥ 2 from the red dot.

Note that if C is a code, and x is a received word, then $C \cap S_1(x)$ consists of all those codewords which differ from x in at most one position. This gives us our decoding strategy!

Algorithm 3.16 — Maximum-Likelihood Decoding Algorithm. The standard ML decoding algorithm for a code $C \subset F^n$ is:

- Suppose we receive the vector $x \in F^n$.
- If $x \in C$, then set $y = x$.
- Otherwise, if $C \cap S_1(x) \neq \emptyset$, then choose $y \in C \cap S_1(x)$.
- If $C \cap S_1(x) = \emptyset$, then repeat with $C \cap S_\ell(x)$, for $\ell = 2, 3, 4, \cdots$ until a codeword y is found.
- Return the decoded value y.

We note some potential issues, however. For example, it would be best if $C \cap S_1 = \{y\}$ was a single codeword; if there is more than one choice, then we don't know which one is the correct codeword.

It turns out that the best way to address this issue is to turn the question around slightly, and consider the open balls centered on the codewords.

■ **Example 3.17** Consider the ternary code $C = \{000, 111, 222\}$. Instead of trying to picture all the codewords on a cube, as in Figure 3.3, we could draw the connections that interest us as a graph for more clarity.

Figure 3.4 is a graph with the elements of the code (in small circles) together with all the elements of the balls of radius 1 centered on codewords (large circles). Edges connect vectors differing in a single bit (Hamming distance 1). Not all vectors of $\mathbb{Z}_3^3$ appear in the graph; one example a vector equidistant from two codewords is shown.

It is clear from this graph that no received word can come from two different codewords through a single error, so the ML decoding algorithm will succeed whenever x results from at most one error. ■

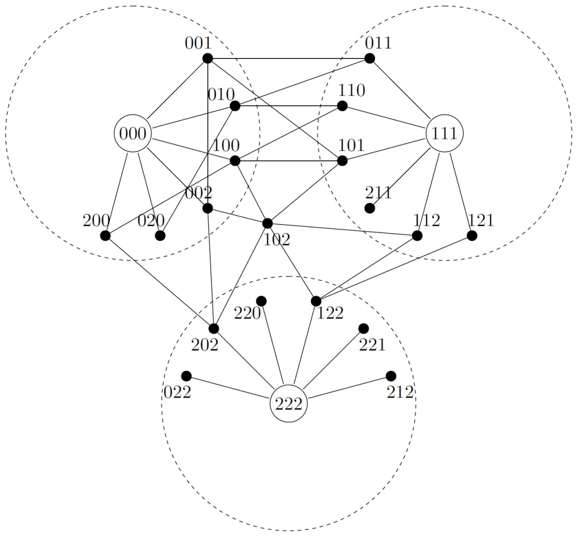


Figure 3.4: A graph of the ternary code in Example 3.17. Image from course notes by Mike Newman.

3.5 The minimum distance of a code

Definition 3.18 Let C be a code in F^n, for some finite field F. The *minimum distance* of C is

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

that is, this is the minimum Hamming distance between distinct codewords.

Although this definition applies to any code, it is most interesting in the case of linear codes, in part due to the following lemma.

Lemma 3.19 If C is a linear code, then its minimum distance can also be computed as

$$d_{min} = \min\{wt(x) \mid x \in C, x \neq 0\}.$$

The proof is left as an exercise. Note that the lemma makes it much simpler to compute the minimum distance of a code: we just need to look at the weights of all the (nonzero) codewords.

■ **Example 3.20** The $(8,7)$ ASCII parity check code has no codewords of weight 1, since if v has only one nonzero entry, say in position j, then $\sum_{i=1}^{8} v_i = v_j \neq 0$. But $10000001 \in C$ and this has weight 2. So the minimum distance is 2. ■

■ **Example 3.21** The $(3,1)$ repetition code $C = \{000, 111\}$ has $d(C) = 3$. ■

■ **Example 3.22** The $(10,9)$ ISBN code C cannot contain any vectors of Hamming weight 1, but we can easily produce elements in C of Hamming weight 2. So $d_{min} = 2$. ■

■ **Example 3.23** The $(6,3)$ code of Example 3.6 has (by inspection of the weights of its nonzero elements), $d_{min} = 3$. ■

How big can d_{min} be?

Theorem 3.24 — Singleton Bound. Let C be an (n,k) linear code. Then $d_{min} \leq n-k+1$.

Proof. Let C be an (n,k) code and set $d = d_{min}$. Define a map

$$\pi : C \to F^{n-d+1}$$

by the projection onto the first $n-d+1$ coordinates. That is,

$$\pi(a_1 \cdots a_n) = a_1 \cdots a_{n-d+1}.$$

This is a linear map; it's just the usual projection map. What is its kernel? Well, if $x \in \ker(\pi)$ then $\pi(x) = 0$. This means that the first $n-d+1$ coordinates of x are all zero. Thus it has at most $d-1$ nonzero coordinates, so $wt(x) < d$; but this is impossible (by Lemma 3.19) unless $x = 0$. Therefore the only element of the kernel is the zero vector, and therefore π is injective.

Now we can bound the size of C. If $|F| = q$, then $|C| = q^k$ and $|F^{n-d+1}| = q^{n-d+1}$. Injectivity implies

$$|C| \leq |F^{n-d+1}|$$

which yields $q^k \leq q^{n-d+1}$ or $k \leq n-d+1$, which gives the result we needed. ■

What makes d_{min} interesting and useful? A good way to think about d_{min} is: if $d = d(C)$ then for any $x \in C$, we have

$$B_d(x) \cap C = \{x\} \qquad (\text{or } S_{d-1}(x) \cap C = \{x\}).$$

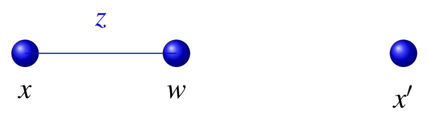
Our next step is to relate this to the error-detecting and error-correcting capabilities of the code.

3.6 Error-correction and error-detection

We begin by giving a concrete definition of error-detection and error-correction. Our perspective: if $x \in C$ is sent but $y \in F^n$ is received, then the difference $z = y - x$ is called the *error vector*. That is, we think of the process of an error occurring in transmission as being realized algebraically as "adding the error vector z". Any vector can thus be an error vector (including 0, which is no error at all); given x, y, z as above, we say x is the codeword, z is the error vector and y is the received word.

Definition 3.25 A code C is said to *detect t errors*, or be t-error detecting, if for all $x \in C$, and all possible nonzero vectors z of weight t or less, $x + z \notin C$.

A code C is said to *correct* **t** *errors*, or be **t**-error correcting, if for all $x \in C$, and all possible vectors z of weight **t** or less, x is closer to $w = x + z$ than is any other codeword in C, that is, $d(x,w) < d(x',w)$ for all $x' \in C$, as illustrated below.



Note that we insist that ALL possible errors of the given weight must be detectable or correctable; in general a code may also detect more than t errors, or correct more than $\mathbf{t}$ errors, in special cases, but we are only interested in the general (worst!) case.

Let us first understand the condition of error-correction better.

Lemma 3.26 A code C can correct $\mathbf{t}$ errors if and only if

$$\forall x \in C, \forall w \in S_{\mathbf{t}}(x), \quad S_{\mathbf{t}}(w) \cap C = \{x\}. \tag{3.2}$$

The setup is illustrated in Figure 3.5.

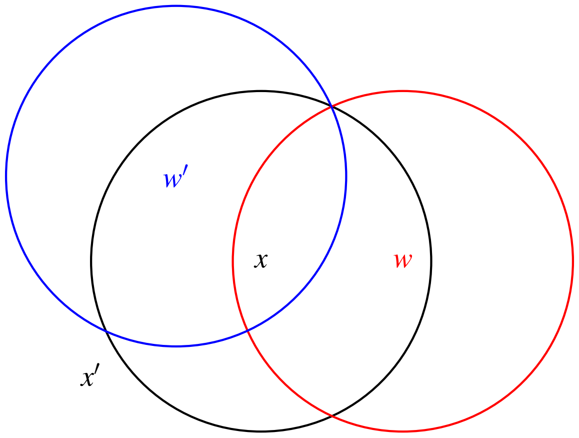
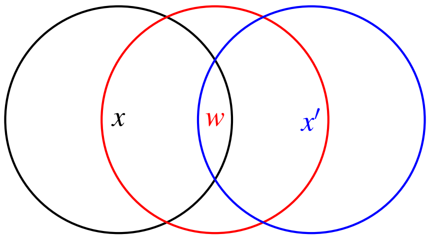


Figure 3.5: The black circle represents the ball of radius $\mathbf{t}$ around a codeword x. The vectors w and w' are in $S_{\mathbf{t}}(x)$. The balls $S_{\mathbf{t}}(w)$ (red) and $S_{\mathbf{t}}(w')$ (blue) contain no other elements of C besides x.

Proof. Suppose first that the condition in the statement labelled (3.2) holds. Let $x \in C$ be arbitrary, and let z be any error vector such that $wt(z) \leq \mathbf{t}$. Then $w = x + z$ is at a distance of at most $\mathbf{t}$ from x. Since $S_{\mathbf{t}}(w) \cap C = \{x\}$, there are no other codewords x' such that $d(x', w) \leq \mathbf{t}$, so x is the unique closest codeword to w. Thus C is $\mathbf{t}$-error correcting.

Now suppose that C can correct $\mathbf{t}$ errors and suppose to the contrary that (3.2) fails. That is, there exists some $x \in C$ and some $w \in S_{\mathbf{t}}(x)$ such that $S_{\mathbf{t}}(w) \cap C \neq \{x\}$. Now since $d(x, w) \leq \mathbf{t}$ and $x \in C$ we know that $x \in S_{\mathbf{t}}(w) \cap C$, so inequality means that there exists another codeword $x' \in C$, $x' \neq x$, such that $x' \in S_{\mathbf{t}}(w)$. So we have $d(x, w) \leq \mathbf{t}$ and $d(x', w) \leq \mathbf{t}$.[3] We illustrate this setup in the following picture.

[3]Note that these inequalities do not imply that $d(x', w) < d(x, w)$, which would be an immediate contradiction.



We can interpret this in two ways:

- If x was the sent word, then $z = w - x$ is the error; since $wt(z) \leq \mathbf{t}$, the fact that C corrects $\mathbf{t}$ errors promises that $d(w,x) < d(w,x')$;
- If x' was the sent word, then $z' = w - x'$ is the error; since $x' \in S_{\mathbf{t}}(w)$, $wt(z') \leq \mathbf{t}$. Thus e-error correction promises instead $d(w,x') < d(w,x)$.

This is a contradiction; no such x' can exist. So (3.2) holds. ■

We can now give a wonderfully simple criterion for error detection and error correction.

Theorem 3.27 Suppose C is an (n,k) code over F. Then

1. C can detect t errors if and only if $t < d(C)$.
2. C can correct $\mathbf{t}$ errors if and only if $\mathbf{t} \leq \dfrac{d(C)-1}{2}$.

Proof. For the first part: A code C can detect t errors if and only if for each $x \in C$, there is no $y \in C$ such that $d(x,y) \leq t$ (that is, t errors aren't enough to turn one codeword into another; they can only turn a codeword into a non-codeword). This latter condition is equivalent to saying that $d_{min} > t$.

For the second part, by the lemma, we will be done if we show that (3.2) holds if and only if $2\mathbf{t}+1 \leq d(C)$.

First suppose that $d(C) \leq 2\mathbf{t}$. Then there exists some $x, y \in C$, $x \neq y$, such that $d(x,y) \leq 2\mathbf{t}$. Construct (exercise) a vector w such that $d(x,w) \leq \mathbf{t}$ and $d(y,w) \leq \mathbf{t}$. Then this w contradicts (3.2). We conclude that if $d(C) \leq 2\mathbf{t}$, C is not $\mathbf{t}$-error correcting. Equivalently, if C is $\mathbf{t}$-error correcting, then $d(C) \geq 2\mathbf{t}+1$.

Conversely, suppose $d(C) \geq 2\mathbf{t}+1$, and let's show that (3.2) holds. So let $x \in C$ and $w \in S_{\mathbf{t}}(x)$. Suppose to the contrary that there exists $y \in C \cap S_{\mathbf{t}}(w)$, $y \neq x$. Then by the triangle inequality (of the Hamming metric) we'd have

$$d(x,y) \leq d(x,z) + d(z,y) \leq \mathbf{t} + \mathbf{t} = 2\mathbf{t},$$

contradicting that the minimum distance of this code is greater than $2\mathbf{t}$. So (3.2) cannot fail (it must hold). ■

■ **Example 3.28** We see that neither our parity check nor our ISBN code have $d_{min} \geq 3$, which is the minimum for correcting at least one error; but they can each detect 1 error since $d_{min} = 2$. On the other hand, our repetition code has $d_{min} = 3$ so $\mathbf{t} = 1$ and $t = 2$. ■

■ **Example 3.29** Our $(6,3)$ code has $d_{min} = 3$ so $\mathbf{t} = 1$, and so this code can correct all single errors. Notice that the rate on this code is $3/6 = 1/2$, which is better than the repetition code, for the same error-correcting capability. ■

We can visualize this theorem with our usual pictures of closed balls. The code is t-error-detecting if one can fit balls of radius t around each codeword without meeting other codewords; the code is $\mathbf{t}$-error correcting if the balls of radius $\mathbf{t}$ centered at all codewords are *disjoint*.

3.7 Hamming bound

Remembering that we are working over finite fields, and that all these sets are finite, leads us to a more interesting relationship between n, k and d_{min} than the Singleton Bound. The following result was proven for the case $p = q = 2$ in 1950 by Richard W. Hamming. Because of the method of the proof, others call it simply the Sphere-packing bound.

Theorem 3.30 — Hamming bound (when $q = p = 2$); Sphere-packing bound (when $q > 2$). If C is an (n,k) linear q-ary code which can correct $\mathbf{t}$ errors, then

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

Proof. We just discussed that C is $\mathbf{t}$-error-correcting iff the closed balls of radius $\mathbf{t}$ centered at elements of C are disjoint. Let's count how many elements of F^n lie in each of these open balls. It is clear that they are all of the same cardinality, so it suffices to look at the zero codeword.

Recall that $S_{\mathbf{t}}(0) = \{x \in F^n \mid d(x,0) \leq \mathbf{t}\}$. Enumerating the vectors in this set according to the number of bits in which they differ from x yields

$$|S_{\mathbf{t}}(0)| = \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{e}(q-1)^{\mathbf{t}}.$$

We have $|C| = q^k$ such balls. Since they are disjoint, we have

$$|C| \times |S_{\mathbf{t}}(0)| \leq |F^n|$$

which gives

$$q^k \left(\sum_{i=0}^{\mathbf{t}} \binom{n}{i} (q-1)^i \right) \leq q^n.$$

■

The case of $q = 2$ is an important one, as it yields a fundamental identity relating $\mathbf{t}$, k and n:

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{\mathbf{t}} \leq 2^{n-k}. \tag{3.3}$$

This arises so frequently (due to the popularity of binary codes) that we call it the Hamming bound.

We now have two inequalities that constrain the dimension k of a linear code of length n and minimum distance d:

- the Singleton bound (Theorem 3.24) which says that $k \leq n - d + 1$; and

- the Hamming bound (Theorem 3.30), which says that

$$q^k \leq q^n \left(\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} (q-1)^i \right)^{-1}.$$

We say that a code meets a bound if the inequality becomes an equality for that code.

Definition 3.31 A code C which meets the Singleton bound, that is, for which

$$d_{min} = n - k + 1$$

(equality) is called a *maximum distance separable* or MDS code. A code which meets the Hamming or sphere-packing bound, that is, for which

$$\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} (q-1)^i = q^{n-k}$$

(equality) is called a *perfect code*.

■ **Example 3.32** Our parity check, ISBN and repetition codes are all MDS since $d_{min} = n-k+1$ in all cases. Since the parity check and ISBN codes have $\mathbf{t} = 0$, they cannot be perfect (since the left hand side of (3.3) is 1). But in fact, the repetition code has $n = 3$, $\mathbf{t} = 1$, $k = 1$ which yields

$$1 + n = 4 = 2^{3-1} = 2^{n-k}$$

so this code is perfect. ■

■ **Example 3.33** Our $(6,3)$ code is not MDS, since it does not meet the Singleton bound (that is, $d_{min} \neq n-k+1$). Is is also not perfect, since the left side of the Hamming bound formula (3.3) adds to 7 whereas the right side is $2^3 = 8$.

In fact, it's immediately clear that *no* $(6,3)$ code could ever be perfect! ■

3.8 Exercises

1. Prove that if $0 \leq p \leq 1$ then $0 \leq H_2(p) \leq 1$. Compute the capacity of a channel if $p = 0$, $p = 1/2$ or if $p = 1$. What does the case $p = 1$ represent and how is this consistent with the answer you get for the capacity of such a channel?
2. Suppose you have a binary symmetric channel with a bit error rate of $p = 0.1$. For a single-error correcting code of length $n = 32$, what is the expected block error rate? What is the maximum rate R of a code on this channel? To what value of k (dimension of the message space) does this correspond?
3. Find a book with a 10-digit ISBN and verify that this number lies in the ISBN code, as defined in Example 3.5. Show that it can detect a single error in any position, as well as a transposition error.
4. Let C denote the ISBN code.
   (i) Prove that if $a \in C$ and b satisfies $d(a,b) = 1$ then $b \notin C$.
   (ii) Prove that if $a \in C$ and b is the result of swapping two digits of a, then $b \notin C$.

(iii) Thinking about how you proved the properties in parts (i) and (ii), explain why the ISBN code is mod 11, instead of the more convenient mod 10. Give examples to justify the superiority of 11 over 10.

5. Show that the set C of all vectors in $\mathbb{Z}_2^n$ of even weight is a subspace of Z_2^n, and is therefore a code. Show that as a subgroup of Z_2^n it is of index two; or equivalently, that its dimension is $n-1$. Hint: over $\mathbb{Z}_2$, the map $x \mapsto wt(x)$ is a homomorphism.
6. Let C be the set of all vectors in $\mathbb{Z}_3^n$ whose weight is even. Show that this is not a subspace of $\mathbb{Z}_3^n$, and so is not a code.
7. Prove Lemma 3.12.
8. (For students who've seen some topology) The open balls defined by the Hamming metric suffice to define the open sets of a topology on F^n. If you have encountered this notion in your other courses, then answer: what is this topology? Does the Hamming metric on $\mathbb{R}^n$ induce the usual Euclidean topology?
9. Explore trying to adapt the Euclidean metric to $\mathbb{Z}_3^n$, that is, by associating elements of $\mathbb{Z}_3$ with real numbers and then using the formula $\|x\| = \sqrt{x_1^2+x_2^2+x_3^2}$ (so that $d(x,y) = \|x-y\|$). What kinds of problems arise when you try to verify the definition?
10. Describe the "shape" of a closed ball of radius 1 in $\mathbb{Z}_p^2$ and $\mathbb{Z}_p^3$ when we embed $\mathbb{Z}_p^3$ into $\mathbb{R}^3$ by mapping elements of $\mathbb{Z}_p$ to integers in $\{0,1,\ldots,p-1\}$ (as in the pictures in this chapter). What is the shape of a closed ball of radius 2?
11. Is $|S_{r+1}(x)| > |S_r(x)|$ always true? Discuss.
12. The Hamming weight is a *norm* that extends the trivial absolute value on the finite field. That is, for all $x,y,z \in F^n$:
    (i) $wt(x) \geq 0$ and $wt(x) = 0$ iff $x = 0$;
    (ii) $wt(cx) = wt(x)$ for any nonzero scalar $c \in F$;
    (iii) $wt(x+y) \leq wt(x) + wt(y)$.

    (But for those of you who have seen more rings and fields: this is **not** the same thing as an algebraic norm.... same spirit, different definition, different uses!)
13. Prove Lemma 3.19.
14. Show that if C is an (n,k) code of minimum distance d, then you can erase any selection of $\ell \leq d-1$ coordinates and the result will be an $(n-\ell,k)$ code. That is, the size of the code does not change. On the other hand, prove that erasing $\ell = d$ coordinates produces an $(n-d,k')$-code with $k' < k$.
15. Suppose that $d(x,y) < 2\mathbf{t}$. Show how to explicitly create w such that w is at a Hamming distance of at most $\mathbf{t}$ from each of x and y.
16. The *Luhn algorithm* was patented in the 1950s as a quick way to create a check digit which would allow the detection of single errors as well as the detection of transposition errors. It is currently used on credit cards as well as Canadian social insurance numbers. The algorithm is: take your card number and reverse it:

$$x_{16}x_{15}\cdots x_3x_2x_1,$$

where each $x_i \in \{0,1,\ldots,9\}$. For each i, define
- y_{2i} = the sum of the digits of $2x_{2i}$;
- $y_{2i+1} = x_{2i+1}$.

Then the card number is a valid codeword iff

$$\sum_{i=1}^{16} y_i = 0 \mod 10.$$

(i) Verify your own credit card number and 9-digit SIN with this algorithm. Keep this calculation private, please, as these numbers should always remain confidential.
(ii) Let C denote the set of valid credit card numbers; this is a subset of $\mathbb{Z}_{10}^{16}$ (for most credit cards). Explain why C cannot be a linear code, as per our definition. Explain further why C is not a subset of a linear code in $\mathbb{Z}_{11}^{n}$, as was the case for 10-digit ISBN numbers.
(iii) Show that the Luhn algorithm will detect all single errors. Find a transposition error which the Luhn algorithm does not detect.

17. Newer ISBN numbers have 13 digits: $x_0x_1\cdots x_{12}$. The checksum is calculated as

$$\sum_{i=0}^{6} x_{2i} + 3\sum_{i=0}^{5} x_{2i+1} \equiv 0 \mod 10.$$

In particular, since we are no longer working modulo 11, the extra symbol X is not required.
(i) Expanding to 13-digit ISBNs instead of 10-digit ISBNs significantly increases the number of valid ISBNs. Suppose instead they had decided to allow X to occur in any digit of a 10-digit ISBN, instead of only the check digit. Estimate the sizes of these three codes (13-digit, 10-digit, expanded 10-digit).
(ii) This code is no longer linear, since we're not working over a field. Show that it is single-error-detecting but not single-error-correcting. Find a also transposition error that it cannot detect.
(iii) Could we have replaced 3 by 2 as a coefficient in the sum? Why 3?

18. This exercise explores some results related to the existence of good codes.
(i) Prove the (named for independent discoverers E. N. Gilbert (1952) and R. R. Varšamov (1957)): the maximum possible size of a code in $\mathbb{F}_q^n$ of minimum distance d is at least

$$\frac{q^n}{\sum_{j=0}^{d-1}\binom{n}{j}(q-1)^j}.$$

This bound would feel meaningless except they went on to prove that there exist linear codes over $\mathbb{F}_q$ satisfying this bound — but by arguing that a random linear code of minimal distance d would satisfy this bound with high probability, which did not yield a construction of such codes.
(ii) Verify if this bound is satisfied by the linear codes we have seen. Codes surpassing this bound are sometimes called capacity-approaching because of the relation to Shannon's channel capacity bound.

4. Towards a systematic construction of codes

How many “different” error-correcting codes are there? How can we classify them, and eventually, figure out which ones are “best" by a given measure? A good place to start is to ask ourselves: when do we consider two codes “different" or “the same"?

4.1 Equivalent codes

We have defined a linear (n,k) code as a k-dimensional subspace of F^n. However, the *category* of codes is not the same as the *category* of subspaces! Namely, any two k-dimensional subspaces are isomorphic; but since we care about the weights of the codewords, we do not consider any two (n,k) codes to be “isomorphic”. We can have fundamentally different codes of the same size, with the same values of n and k!

But to classify codes, we need a precise definition of when we consider them to be equivalent[1]. Recall that the group of permutations $\mathscr{S}_n$ acts on F^n by permuting the coordinates (see Appendix A.4).

Definition 4.1 Let C_i be an (n_i,k_i) code over F_i, for $i=1,2$. Then C_1 is *equivalent* to C_2, written $C_1 \sim C_2$, if $n_1=n_2$, $k_1=k_2$, $F_1=F_2$ and there exists a permutation $\sigma \in \mathscr{S}_n$ such that $\sigma(C_1)=C_2$.

■ **Example 4.2** Consider the following three $(4,2)$ binary codes.

$$C_1 = \{0000, 1100, 0011, 1111\}$$
$$C_2 = \{0000, 1010, 0101, 1111\}$$
$$C_3 = \{0000, 1100, 0110, 1010\}$$

The permutation (23) takes C_1 onto C_2 (and vice versa, since (23) has order 2). But since any permutation of 1111 is 1111, there is no permutation taking C_1 (or C_2) to C_3. Therefore C_1 and C_2 are equivalent, but C_3 in not equivalent to either of them. ■

[1]We will not use “isomorphic" and reserve that for vector spaces.

Lemma 4.3 If $C_1 \sim C_2$ then $d(C_1) = d(C_2)$.

Proof. Let $\sigma \in S_n$. Then since σ just permutes the coordinates of a vector c, in particular c and $\sigma(c)$ have the same number of nonzero coordinates. Thus $wt(c) = wt(\sigma(c))$. Applying this to elements of least positive weight in each code yields the lemma, since these weights are the minimum distances of their respective codes. ■

Example 4.2 shows that the converse is not true.

R Many authors use a broader definition of *scaled equivalence* for q-ary codes: $C_1 \sim C_2$ if there is a permutation matrix P and an invertible diagonal matrix D such that $\{DPc \mid c \in C_1\} = C_2$. For binary codes, this is exactly equivalence as defined above.

■ **Example 4.4** The two ternary codes $C_1 = \{00, 11, 22\}$ and $C_2 = \{00, 12, 21\}$ are not equivalent, but they are scaled-equivalent since with

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

we have $DC_1 = C_2$. ■

4.2 Systematic codes

Now that we have a notion of equivalence of codes, we may proceed towards a classification. The first step is to introduce the generator matrix of a code.

Definition 4.5 Let C be a linear (n,k) code over the field F. Choose an encoding function, that is, an injective linear transformation

$$\varphi \colon F^k \to F^n$$

whose image is the subspace C. Then the standard matrix of this transformation is called a *generator matrix* for C.

■ **Example 4.6** Binary repetition code: The map sends 1 to 111 so and the corresponding generator matrix is

$$G = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

. ■

The encoding function takes a message, which is a vector in F^k, and converts it into a codeword $c \in C \subseteq F^n$. In terms of the generator matrix, this just means that the message m is converted into the codeword Gm, where this is matrix multiplication.

! In the coding literature, it is very common to transpose the generator matrix, so that G^T is $k \times n$ rather than $n \times k$. Then to encode a *row vector* m^T, you multiply by the matrix on the *right*, that is, compute $m^T G^T$ instead. Since $m^T G^T = (Gm)^T$, this comes out to the same thing.

Lemma 4.7 If C is a linear code with generator matrix G, then the columns of G form a basis for C. Conversely, any basis for C determines a generator matrix G.

Proof. This follows from the definition and linear algebra. Let $\varphi\colon F^k \to F^n$ be an injective linear map with matrix G. Then the columns of G are precisely $\{\varphi(e_i) \mid 1 \le i \le k\}$, where $\{e_i \mid 1 \le i \le k\}$ is the standard basis for F^k. Since φ is injective, this is a basis for the image $\varphi(F^k) = C$.

Conversely, given a basis $\{c_1, \dots, c_k\}$ for C, specify a linear transformation $\varphi\colon F^k \to F^n$ by defining $\varphi(e_i) = c_i$ for each $1 \le i \le k$ and extending by linearity to a function on F^k. Then the corresponding matrix G has columns equal to this basis. ■

■ **Example 4.8** Binary parity check $(3,2)$ code: $C = \{000, 101, 011, 110\}$. Then we have several possible generator matrices, including $G_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$ and $G_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$. We note that G_1 has the nice property that a message x_1x_2 is encoded as x_1x_2p where p is the parity check bit, that is, the message occurs as the first part of the codeword. This is not true of G_2, where for example 11 is encoded as $G\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 011$. ■

Generator matrices in *standard form*

$$\begin{bmatrix} I_k \\ A \end{bmatrix}$$

for some $n-k \times k$ matrix A, are particularly appealing because for any message $u \in F^k$, the corresponding codeword is

$$Gu = \begin{bmatrix} u \\ Au \end{bmatrix}.$$

Definition 4.9 A linear code is called a *systematic code* if it has a generator matrix which is in standard form.

■ **Example 4.10** One generator matrix of the $(6,3)$ code of Example 3.6 is

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

so this code is systematic. ■

■ **Example 4.11** The $(3,2)$ code $C_2 = \{000, 101, 110, 111\}$ defined earlier is not systematic, since there are no codewords starting with the string 01. This is impossible in a $(n,2)$ systematic code. ■

Theorem 4.12 Every linear code is equivalent to a systematic code.

Proof. Since equivalence is defined by permuting coordinates, we see that what we have to show is that there is a basis for C and a set of k coordinates such that each basis vector has a 1 in exactly one of these coordinates, and zeros in the rest. That is, we want a really nice, simple basis for C — which is exactly what we learned to do in linear algebra, using row reduction. Row reduction reduces a matrix in such a way that the rowspace is unchanged, but the resulting basis for the rowspace is as simple as possible (in precisely the sense that we require here).

So let G be any generator matrix for C. Then the rowspace of G^T is equal to C. Row reduce G^T using Gauss-Jordan elimination. Since $\dim(C) = k$, the rank of G^T is k, and so in the reduced row echelon form R^T of G^T, there are precisely k leading ones, with zeros above and below. Since the rowspace of G^T equals that of R^T, we deduce that R is another generator matrix for C.

Now let $\sigma \in \mathscr{S}_n$ be such that the k leading ones of R^T are rearranged to be the indices $1, 2, \ldots, k$. Then $\sigma(C)$ is a systematic code. ■

■ **Example 4.13** Let's apply the argument of this proof to find a systematic code equivalent to

$$C = \{00000, 10201, 20102, 02120, 01210, 11111, 22222, 21012, 12021\}.$$

First, we choose a basis; this is a $(5,2)$ ternary code so any two vectors which are not scalar multiples will do. We could look carefully and make a good choice; but for the sake of argument let us choose

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

Now row reduce G^T:

$$\begin{bmatrix} 2 & 0 & 1 & 0 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \sim R_1 + R_2 \to R_2; 2R_1 \sim \begin{bmatrix} 1 & 0 & 2 & 0 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

which is (as expected) another basis for the code — and in standard form, no less.

If instead we had begun with

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

then we might have realized that rows 2 and 3; or rows 3 and 4, satisfy the requirements for giving a standard matrix after permutation. If we choose $\sigma = (1324)$, for example, then

$$\sigma(G) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 2 & 2 \\ 2 & 2 \end{bmatrix}$$

which is a standard generator matrix, not for C, but for the equivalent code $\sigma(C)$. ■

This gives us one answer to the classification question we began with: every equivalence class of codes contains at least one matrix in standard form. So if we wish to generate many different codes, it suffices to generate many standard matrices of the size we desire.

Unfortunately, none of this discussion has shed any light on the minimum distance of such codes. This is our next goal.

4.3 Exercises

1. Imagine we wanted to exhaustively search among all possible codes of length n and dimension k to find the one with the highest d_{min}. To get a sense of the size of this calculation, show that the number of 3-dimensional binary codes of length n is

$$\frac{(2^n-1)(2^{n-1}-1)(2^{n-2}-1)}{21}.$$

(Hint: count the number of ways of choosing 3 linearly independent vectors, then quotient by the number of bases of a 3-dimensional subspace.) Find a formula for the number of (n,k) codes.
2. Prove that if $C_1 \sim C_2$, then $C_2 \sim C_1$. (This needs proving since the definition is asymmetric.) In fact, show that $\sim$ is an equivalence relation.
3. Prove that a linear map $\varphi\colon F^k \to F^n$ is injective if and only if the image under ϕ of any basis of F^k is a basis for $im(\varphi)$.
4. Let us recall why the rowspace of a matrix A equals that of its reduced row echelon form R. Row reducing A implies multiplying A on the left by a matrix B; so $BA = R$. The rowspace of A is the span of the rows of A, or equivalently, the span of the columns of A^T; thus it is the image of the linear transformation corresponding to A^T. Explain why the image of A^T (that is, the rowspace of A) equals that of R^T; this is linear algebra.
5. Apply the argument of the proof of Theorem 4.12 to the code $C = \{0000, 1100, 0011, 1111\}$ to determine an equivalent systematic code C and generator matrix G in standard form.
6. Find the minimum distance of the code with generator matrix

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

In general, is it true that $d(C)$ is equal to the minimum weight of the column vectors of a generator matrix G? Discuss.
7. Classify all $(3,1)$ binary codes up to equivalence, as follows.
   (i) Begin by finding out how many there are, and writing down a generator matrix for each of them.
   (ii) Then write down all standard generator matrices for systematic binary $(3,1)$ codes.

(iii) By our theorem, every code is equivalent to one of these; but it is also possible for two standard matrices to correspond to equivalent systematic codes. Determine all equivalence classes, and which equivalence classes have a unique systematic representative.

8. For the codes of the preceding exercise, verify that d_{min} is an invariant of the equivalence classes.
9. Give an estimate of (or: lower and upper bounds for) the number of (n,k) codes up to equivalence. (This question is open-ended, inviting you to use whatever tools you have at hand, or to specialize to any particular sub-case you find amenable to analysis.)

4.4 The dual code

Although generator matrices permit us to exhaustively enumerate all codes up to equivalence, they don't particularly answer the question: how "good" is the code (*i.e.* what is d_{min}?). For this, and other important questions, we need a related tool.

Definition 4.14 Let C be an (n,k) linear code over F. Then the *dual code* is the set

$$C^{\perp} := \{v \in F^n \mid v \cdot w = 0 \ \forall w \in C\}$$

where $v \cdot w = \sum_{i=1}^{n} v_i w_i$.

■ **Example 4.15** Let $C = \{000, 111\}$ be the binary repetition code. Then

$$\begin{aligned} C^{\perp} &= \{x \in \mathbb{Z}_2^3 \mid x \cdot w = 0 \forall w \in C\} \\ &= \{x \in \mathbb{Z}_2^3 \mid x \cdot 111 = 0\} \\ &= \{x \in \mathbb{Z}_2^3 \mid x_1 + x_2 + x_3 = 0\} \\ &= \{000, 101, 011, 110\} \end{aligned}$$

which is the binary $(3,2)$ parity check code. ■

We can show, using the bilinearity of the dot product, that $C^{\perp}$ is a subspace of F^n, hence itself a code (exercise). To determine its size, we need to develop a bit further the theory of bilinear forms over arbitrary fields.

4.5 Dot products over a finite field

We are familiar with the dot product over $\mathbb{R}$; it is a positive definite inner product there. So if W is a subspace of $\mathbb{R}^n$, then $W^{\perp}$ is the orthogonal complement of W in $\mathbb{R}^n$, whence $W^{\perp}$ is a subspace of dimension $n-k$ that intersects W only in $\{0\}$.

However, the dot product is only a symmetric bilinear form over F^n. One of the major differences is the existence of non-zero self-orthogonal vectors over a finite field. For example, in the vector space $(\mathbb{Z}_2)^3$, we have

$$110 \cdot 110 = 0.$$

Therefore, in general it will not be true that $W \cap W^{\perp} = \{0\}$. Although it isn't quite right to think of vectors v and w as being orthogonal if $v \cdot w = 0$, this is the terminology we inherit from $\mathbb{R}$.

More accurately, we can think of the dot product as giving us a way of mapping vectors in F^n to vectors in the dual vector space $(F^n)^*$ of linear functionals on F^n[2], as follows.

Lemma 4.16 The map

$$\psi \colon F^n \to (F^n)^*$$

defined by: for each $v \in F^n$, we declare $\psi(v)$ to be the linear functional on F^n given by

$$\psi(v)(w) = v \cdot w.$$

This is a well-defined linear isomorphism.

Proof. We leave linearity (in both v and w) and well-definedness as an exercise.

Since $\dim(F^n) = \dim(F^n)^* = n$ and ψ is linear, it suffices to show that ψ is injective. Note that $\psi(v) = 0$ if and only if $\psi(v)(w) = 0$ for all $w \in F^n$. But $\psi(v)(e_i) = v_i$, the ith coordinate of v. So this is zero for all e_i if and only if $v = 0$. So $\ker(\psi) = \{0\}$ meaning ψ is injective, hence an isomorphism. ■

This is not the only isomorphism between these two spaces (by any means!). Let $B = \{v^1, v^2, \dots, v^n\}$ be a basis of F^n. Then we can define $B^* = \{f^1, f^2, \dots, f^n\}$ of $(F^n)^*$ by setting each f^j to be the unique linear functional which takes value 1 on v^j and 0 on each v^i, $i \neq j$. The map $\rho_B \colon F^n \to (F^n)^*$ which sends v^i to f^i for each i is another isomorphism of the two spaces.

These isomorphisms hold for any field F. Over $\mathbb{R}$, however, if B is an orthonormal basis then $\psi(B) = B^*$. The preponderance of orthonormal bases then makes the rest easy. Even though we do not usually have orthonormal bases over finite fields, nonetheless, what we have for arbitrary F is enough.

Proposition 4.17 Suppose $W \subseteq F^n$ is a k-dimensional subspace and set

$$W^\perp = \{x \in F^n \mid x \cdot w = 0 \quad \forall w \in W\}.$$

Then

1. $W^\perp$ is a subspace of F^n of dimension $n - k$.
2. If H is a matrix whose rowspace is W, then its nullspace is $W^\perp$.
3. $(W^\perp)^\perp = W$.

Proof. That $W^\perp$ is a subspace is an exercise; let us show it has the given dimension.

Choose a basis $B_W = \{v^1, v^2, \dots, v^k\}$ for W and extend it to a basis $B = \{v^1, v^2, \dots, v^k, v^{k+1}, \dots, v^n\}$ for F^n. Let $\rho_B(B) = B^* = \{f^1, \dots, f^n\}$ be the dual basis. For each $j = k+1, \dots, n$, set

$$w^j = \psi^{-1}(f^j)$$

and define $W' = span\{w^{k+1}, \dots, w^n\}$. Since ψ is an isomorphism and the f^j are linearly independent, $\dim(W') = n - j$.

[2]This is where the term "dual codes" comes from.

Note that by definition, $\psi(w^j) = f^j$ so $w^j \cdot v = f^j(v)$ for all $v \in V$. Since all these functions f^j, for $j \in \{k+1, \dots, n\}$, are zero on W, this means each w^j is orthogonal to every vector in W, so lies in $W^\perp$. This gives the inclusion $W' \subseteq W^\perp$.

Conversely, suppose $x \in W^\perp$. Then $x \cdot w = 0$ for all $w \in W$, implying that $\psi(x)(w) = 0$ for all $w \in W$. Write $\psi(x) = \sum_{i=1}^n c_i f^i$ for some coefficients c_i. Since for all $j \in \{1, \dots, k\}$ we have

$$0 = \psi(x)(w^j) = \sum_{i=1}^n c_i f^i(w^j) = \sum_{i=1}^n c_i \delta_{ij} = c_j,$$

we infer that $\psi(x) \in \operatorname{span}\{f^{k+1}, \dots, f^n\}$ so $x \in W'$. Thus $W' = W^\perp$ and this space has dimension $n-k$.

For the second part, note that if H is a matrix whose rowspace is W, then the rows h_i of H span W. The nullspace is therefore

$$\operatorname{Null}(H) = \{v \mid Hv = 0\} = \{v \mid h_i \cdot v = 0 \forall i\} = W^\perp.$$

(Some intermediate steps are left as an exercise.)

We leave the proof of the third part as a fun exercise. ■

Note that if a matrix A represents the linear transformation T, then $\operatorname{Null}(A) = \ker(T)$. We thus also sometimes write $\ker(A)$ for the nullspace of A.

4.6 Parity check matrices

Definition 4.18 Let C be a linear (n,k) code. Then any $(n-k) \times n$ matrix H such that $C = \ker(H)$ is called a *parity check matrix* for C.

Note that since $\dim(C) = k$, this condition forces a parity check matrix for C to have rank equal to $n-k$.

By Proposition 4.17, if C is an (n,k) code, then the dual code $C^\perp$ is an $(n, n-k)$ code. This gives us an immediate way to generate a parity check matrix for C.

Proposition 4.19 Let C be a linear (n,k) code. Denote by $G^\perp$ a generator matrix for $C^\perp$. Then $H := (G^\perp)^T$ is parity check matrix for C.

We leave the proof as an exercise.

■ **Example 4.20** Let C be the $(3,1)$ binary repetition code. By Example 4.15 we know $C^\perp$ and see that may take

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

so that a parity check matrix for C is

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

By construction, the rows of H span $C^{\perp}$; in particular, $C = \mathrm{Null}(H)$. ■

The key property of the parity check matrix is that it gives a very easy way to decide if $w \in C$ or not. It is so important that we'll state it as a theorem, even though it is just a restatement of the definition.

Theorem 4.21 Let C be a code with parity check matrix H. Then $w \in C$ if and only if $Hw = 0$.

For systematic codes, it's even easier to create a parity check matrix. Suppose C is a systematic code and $G = \begin{bmatrix} I_k \\ A \end{bmatrix}$ is a generator matrix for C. Then

$$H = \begin{bmatrix} -A & I_{n-k} \end{bmatrix}$$

is a parity check matrix for C since the equality

$$HG = \begin{bmatrix} -A & I_{n-k} \end{bmatrix} \begin{bmatrix} I_k \\ A \end{bmatrix} = \begin{bmatrix} -A + A \end{bmatrix} = 0$$

shows that the columns of G lie in $\ker(H)$; thus $C \subset \ker(H)$ and by dimension we must have equality.

■ **Example 4.22** Suppose C is a binary code with generator matrix

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

This is in standard form so a parity check matrix for C is

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

■

■ **Example 4.23** Suppose C is a ternary code with generator matrix

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

This is in standard form. Recall that $-2 = 1$ and $-1 = 2$ over $\mathbb{Z}_3$, so a parity check matrix is

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

■

4.7 Exercises

1. The oddities are not restricted to the binary field: construct self-orthogonal vectors in $\mathbb{Z}_3^2$, $\mathbb{Z}_3^3$ and $\mathbb{Z}_5^n$, for any $n > 1$.
2. (If you have taken a course in Number Theory) Let p be a prime number. Construct a self-orthogonal vector in $\mathbb{Z}_p^n$, for any $n > 1$.
3. Show that the dot product is bilinear. Use this to show that $W^\perp$ is a subspace of F^n, that $\psi(v)$ is in $(F^n)^*$, and that ψ is a linear map.
4. Prove that if $\{w^1, \cdots, w^\ell\}$ span W then $W^\perp = \{v \mid w^i \cdot v = 0 \quad \forall i\}$. That is, it suffices to check orthogonality on a spanning set. This was needed in the proof of Proposition 4.17.
5. Show that $(W^\perp)^\perp = W$.
6. Prove Proposition 4.19.
7. Find a generator matrix and a parity check matrix for the ASCII $(8,7)$ binary code.
8. Let $C = \{000, 111, 222\}$, a ternary $(3,1)$ repetition code. Find a generator matrix for C in standard form, and the corresponding parity check matrix. Find another, distinct, parity check matrix.
9. Give a generator matrix G for C and a generator matrix $G^\perp$ for the the dual code $C^\perp$, where C is the $(6,3)$ code of Example 3.6.
10. Row reduce the following matrix to reduced row echelon form in two cases: (a) supposing the field over which we are working is $\mathbb{Z}_5$, and (b) if the field is $\mathbb{Z}_7$. Compare the rank and nullity of the matrix in the two cases:
$$A = \begin{bmatrix} 2 & 1 & 3 & 4 & 2 \\ 2 & 4 & 1 & 3 & 0 \\ 1 & 1 & 0 & 3 & 2 \\ 0 & 2 & 3 & 0 & 3 \end{bmatrix}$$
11. If G is a generator matrix for C, and H is a parity check matrix, then show that $HG = 0$. Is $GH = 0$? Suppose H' is another matrix, of any size such that $H'G$ is well-defined and $H'G = 0$. Do the rows of H' span the dual code to C? Does H' satisfy Theorem 4.21?
12. Suppose $C \sim C'$, via the permutation σ. Then if G is a generator matrix for G, then we can choose $G' = P_\sigma G$ as a generator matrix for C', where P_σ is the $n \times n$ permutation matrix representing the permutation σ. If H is a parity check matrix for C, what is a parity check matrix for C'? Prove your answer.
13. We can define *scaled equivalence* of q-ary codes as follows: $C \equiv C'$ if there exists a permutation matrix P and an invertible diagonal matrix D such that $C' = \{DPc \mid c \in C\}$. Give an expression for a generator matrix and a parity check matrix for C' in terms of D, P, G and H (where G is a generator matrix for C and H is a parity check matrix for C).

5. Decoding

Let us take a moment to consider how to efficiently decode a message. Given a received word v, we need to identify the codeword c which is closest to v. The method we have used thus far — comparing v to each c and choosing the one of minimum distance — is quite slow if the code is large. For an (n,k) q-ary code, we would need to calculate q^k differences, each with n bits, for a total of nq^k operations — plus storage and sorting of the answer.

The standard answer in computer science to "what's efficient?" is : use a look-up table. Each possible received word indexes an entry in the table, and that entry determines the decoding. So far, so good.

But how do we create such a table? There are q^n possible received words; decoding each one ahead of time involves $q^n(nq^k)$ operations, which isn't horrible for off-line use, but is pretty tedious nonetheless. There must be a better way.

5.1 The idea : cosets

Let C be an (n,k) q-ary linear code over F. Then C is a subspace, and in particular a subgroup, of F^n. Recall that a *coset* of the subgroup C of F^n is a set of the form

$$x+C = \{x+c \mid c \in C\}.$$

Here are some basic properties of cosets of a subgroup.

Lemma 5.1 Suppose C is a subgroup of a finite group V. Then

1. for any $v \in V$: $v+C = C$ if and only if $v \in C$, in which case we call this the *trivial coset*;
2. $|v+C| = |C|$ for all $v \in V$;

3. if $v, w \in V$ then

$$(v+C) \cap (w+C) = \begin{cases} v+C & \text{if } w \in v+C \\ \emptyset & \text{otherwise;} \end{cases} \tag{5.1}$$

4. the number of distinct cosets of C in V is $[V:C] = |V|/|C|$.

It follows that V is the disjoint union of the distinct cosets of C.

Proof. We only prove (5.1), and leave the rest as an exercise.

Let $v, w \in C$. Suppose first that $w \in v+C$. This means there exists a $c \in C$ such that $w = v+c$. But then $v = w-c$, and $-c \in C$ because C is a subgroup. So $v \in w+C$.

On the other hand, suppose $w \notin v+C$, and suppose to the contrary that there exists some $z \in (v+C) \cap (w+C)$. Then this z can be written as $z = v+c$ for some $c \in C$ as well as $z = w+c'$ for some $c' \in C$. But then we have $w = v+(c-c')$. Since C is a subgroup, $c-c' \in C$, so this says $w \in v+C$, a contradiction. So no such z exists; the intersection is empty. ■

Let us illustrate the lemma with an example that in which you can easily verify the calculations and conclusions.

■ **Example 5.2** If $C = \{000, 111\}$ then its nontrivial cosets in $V = \mathbb{Z}_2^3$ are

$$\begin{aligned} 100+C &= \{100, 011\} = 011+C \\ 010+C &= \{010, 101\} = 101+C \\ 001+C &= \{001, 110\} = 001+C \end{aligned}$$

So there are four cosets in all, each with two elements. We see directly that every vector in $\mathbb{Z}_2^3$ lies in exactly one of these cosets. ■

Thus, if C is an (n,k) code, then by the lemma, F^n is the disjoint union of cosets

$$F^n = \bigcup_{e \in R} (e+C)$$

where R is a set of *coset representatives*; such a set R is not unique. We have that $|R| = |F^n|/|C| = q^{n-k}$.

■ **Example 5.3** Consider $C = \{000, 111\}$, the binary repetition code. By Example 5.2, we can enumerate the cosets as

- $000+C = C$
- $100+C = \{100, 011\}$
- $010+C = \{010, 101\}$
- $001+C = \{001, 110\}$

so this corresponds to a choice of $R = \{000, 100, 010, 001\}$. These representatives R that we chose are in fact the smallest vectors in each coset. We call them *coset leaders* because they define the error pattern which is most likely to have occurred if our received word is either of the vectors in that coset (check!). ■

5.2 The Standard Array and Coset Leaders

By Lemma 5.1, any element of a coset is a representative for the coset. Therefore, for each coset of C in F^n, we may always choose as representative a *coset leader*, which is an element e of that coset with minimal Hamming weight. (It may not be unique.)

Now create a table, called the *standard array*, where the rows are the cosets of C, with the coset leaders as the first column.

■ **Example 5.4** Suppose $C = \{00000, 00111, 11100, 11011\}$. This is a linear $(5,2)$ binary code, with $d_{min} = 3$. To form the standard array, we simply start with vectors of weight 1; since $d_{min} > 1$, none of these are in C; since $d_{min} > 2$, no two of these vectors could be in the same coset.

00000	00111	11100	11011
10000	10111	01100	01011
01000	01111	10100	10011
00100	00011	11000	11111
00010	00101	11110	11001
00001	00110	11101	11010
10010	10101	01110	01001
10001	10110	01101	01010

Once we'd exhausted the weight one vectors, we chose any vectors of weight 2 which did not appear in any previous coset. However, since $d_{min} < 5$, the coset leaders are no longer unique. ■

Lemma 5.5 If $v \in e + C$, where e is the coset leader, then v can decode as $v - e$, and this is the unique closest codeword to v if every other element of $e + C$ has strictly higher Hamming weight.

Proof. Since $v \in e + C$, we know $v - e \in C$, so it is a codeword, and the distance between v and $v - e$ is $wt(e)$. If $c' \in C$ is any other codeword, then $v - c' \in v + C = e + C$. By our choice of coset leader, the weights of all elements of the coset are at least the weight of the coset leader, so $wt(v - c') \geq wt(e)$. Thus $v - e$ is a closest codeword. If e is the unique lowest weight vector in its coset, then $v - e$ is the unique closest codeword to v. ■

We can think of coset decoding as sorting the received words by the most common patterns of errors.

R Suppose our code is e-error correcting. If our received word is the result of more than e errors, then it probably cannot be decoded uniquely. Therefore, in some applications, we may prefer not to try to decode it at all, but rather ask for it to be retransmitted. Under these circumstances, it suffices to build the standard array only for coset leaders of weight e and less, which is very easy to do. Any vector not occurring in the table is the result of too many errors.

5.3 Syndromes

A useful by-product of how the parity check matrix easily identifies elements of the code is the following.

Proposition 5.6 Let C be a code with parity check matrix H. Then $Hw = He$ if and only if w and e lie in the same coset of C.

The proof is left as an exercise.

Definition 5.7 Let C be an (n,k) code over F with parity check matrix H. Then for any $w \in F^n$, the element $Hw \in F^{n-k}$ is called the *syndrome* of w.

A consequence of the proposition is that syndromes give us a more efficient implementation of coset decoding. Instead of computing the standard array of a code (which has q^n entries), we just need to compute the syndromes of all the coset leaders (of which there are q^{n-k}, or sometimes fewer, if we choose to store only the syndromes of errors we can correct), and instead of searching for w in the entire standard array, we search for Hw in the set of syndromes. Then to decode w we compute $w-e$, as before.

Note that if we store the pair (e, He) for all coset leaders e, then this table has $2q^{n-k}$ entries, which is in general much smaller than the size of the standard array. The cost of this decrease in storage is one matrix product Hw for each call to the decoder.

■ **Example 5.8** Consider the code C and parity check matrix H of Example 4.22:

$$H = \left[\begin{array}{cc|ccc} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

We compute some of its syndromes: $H0 = 0$,

$$He_1 = \begin{bmatrix}1\\1\\0\end{bmatrix}, He_2 = \begin{bmatrix}1\\0\\1\end{bmatrix}, He_3 = \begin{bmatrix}1\\0\\0\end{bmatrix}, He_4 = \begin{bmatrix}0\\1\\0\end{bmatrix}, He_5 = \begin{bmatrix}0\\0\\1\end{bmatrix},$$

which are in fact just the 5 columns of H, since our coset leaders are just the standard basis vectors in this case.

Now the set of syndromes must equal all of $\mathbb{Z}_2^3$, since H is of full rank $n-k$ so its image is all of $F^{n-k} = F^3$. We don't have them all; we are missing some syndromes, namely 011 and 111. But the beauty of this parity check matrix is that it's easy to find the missing coset leaders:

$$\begin{bmatrix}0\\1\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} = He_4 + He_5 = H(e_4 + e_5)$$

so that this is the syndrome of the coset leader 00011. Similarly we deduce that the syndrome 111 corresponds to the coset of $e_1 + e_5 = 10001$, which is the coset leader since it has weight 2 (and all cosets with coset leaders of weight 1 have already been accounted for).

So if we receive $w = 10111$, we compute $Hw = 001$, which is the syndrome of $e_5 = 00001$. Therefore we decode w as $w - e = 10110$, which is clearly in C.

If we received $w = 10101$, then $Hw = 011$, which is the syndrome of 00011, so we would decode it as 10110 also. However, there is another weight 2 error vector with syndrome 011, namely $e = 11000$; if we had chosen it as our coset leader instead we'd have decoded w as 01101. This ambiguity is a reflection of the fact that C is only single error correcting, and more than one error occcurred in the transmission which led to w. ■

The binary case is a bit special: the one-bit errors are exactly the n standard basis vectors. Over a field with $q > 2$ elements, there are $(q-1)n$ vectors with Hamming weight 1. But this does not significantly add to the complexity, as we can see from an example.

■ **Example 5.9** Let C be the ternary code from Example 4.23, whose generator and parity check matrices are

$$G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \text{and} \qquad H = \begin{bmatrix} 1 & 1 & | & 1 & 0 & 0 \\ 0 & 2 & | & 0 & 1 & 0 \\ 2 & 0 & | & 0 & 0 & 1 \end{bmatrix}.$$

We have previously seen (in an exercise) that $d(C) = 3$ so this is a single-error correcting code. What are all the possible single-bit errors? We can list them:

$$10000, 20000, 01000, 02000, 00100, 00200, 00010, 00020, 00001, 00002.$$

In the previous example, which was a binary code, the syndromes corresponding to single bit errors were precisely the columns of H; here, the error e_i gives He_i equal to the ith column of H, while the syndrome of $2e_i$ is $H(2e_i) = 2He_i$ is twice the i column of H.

Suppose we receive the codeword $w = 11201$. Then $Hw = 120$ which is just the second column of H. Thus the error was $e = e_2 = 01000$, and we decode $w - e = 10201$; it's a codeword.

Now suppose $w = 00210$. Then $Hw = 210$, which is twice the second column of H. Thus the error was $e = 2e_2 = 02000$, and we decode $w - e = 00210 - 02000 = 01210$; it's a codeword.

Now suppose $w = 11111$. Then $Hw = 000$, so w is a codeword, and there was no error.

Finally, if we receive $w = 00110$, then $Hw = 110$, which is not 0, or a column of H, or twice a column of H. Thus it is not a codeword, nor is it the result of a single error in transmission. ■

We can summarize some of what we discovered in these examples in a lemma.

Lemma 5.10 Suppose H is a parity check matrix for a code C such that H has no zero columns. If the syndrome of a received word w is equal to a times the ith column of H, then we decode

$$w - ae_i$$

where $e_i \in F^n$ is the ith standard basis vector.

The proof of this lemma amounts to showing that $w - ae_i \in C$ and recognizing that $d(w, w - ae_i) = 1$ (unless $a = 0$, in which case $w = w - ae_i \in C$) so it is a codeword of minimum distance from w (exercise).

However, what is more important for our purposes is the following theorem, which gives us another tool for finding d_{min} of a code (besides writing out all the codewords!).

Theorem 5.11 The minimum distance d of a linear code C is the size of the smallest dependent set of columns of H.

What this means is: given a code C, write down a parity check matrix H. If H has a zero column, then $d(C) = 1$, since $\{0\}$ is a dependent set of columns of H. If two columns of H are scalar multiples of one another, then $d(C) = 2$. But if no two columns of H are scalar multiples of one another, then the minimum set of dependent columns is at least 3, so $d(C) \geq 3$, and C is at least single error correcting!

Proof. Let h^i denote the ith column of H. If $d(C) = d$ then there exists a $c \in C$ such that $wt(c) = d$. Since $c \in C$, we have $Hc = 0$ by Theorem 4.21. Writing this out as an equation on the columns of H gives $\sum c_i h^i = 0$; the number of nonzero coordinates of c is the number of vectors appearing in this dependence relation. So there exists a set of d dependent vectors among the columns of H.

Conversely, given any dependence relation $\sum c_i h^i$ on the columns of H, we can write this as $Hc = 0$ for the corresponding vector of coefficients c. By our fundamental theorem about parity check matrices (Theorem 4.21), we deduce that $c \in C$. Further: $wt(c)$ is exactly the number of terms appearing in this relation. ■

▪ **Example 5.12** The binary $(3,1)$ repetition code has parity check matrix

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

Every pair of columns is linearly independent, but since the columns lie in $\mathbb{Z}_2^2$, the three must be dependent. Thus $d_{min} = 3$, as we knew. ▪

▪ **Example 5.13** The binary $(3,2)$ parity check code has parity check matrix $H = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ since it's the dual of the repetition code. The first two columns (indeed, any pair of columns) are linearly dependent, so $d_{min} = 2$, as we knew. ▪

▪ **Example 5.14** Consider a ternary code with basis $\{1010, 0101\}$. So we can take $G = \begin{bmatrix} I_2 \\ I_2 \end{bmatrix}$, which is in standard form. Thus

$$H = \left[\begin{array}{cc|cc} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{array}\right]$$

is a parity check matrix for C. The first and third columns of H are dependent, so $d_{min} = 2$. ▪

▪ **Example 5.15** Consider the following parity check matrix of our $(6,3)$ binary code from Example 3.6:

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

No two columns are scalar multiples of one another, but the first plus the second equals the third, so the first 3 are dependent. Thus $d_{min} = 3$. ■

5.4 Exercises

1. Prove Proposition 5.6: If C is a code with parity check matrix H, then $Hw = He$ if and only if w and e lie in the same coset of C.
2. Prove Lemma 5.10.
3. Create a standard array and identify coset leaders for the $(6,3)$ code of Example 3.6.
4. Create a standard array and identify coset leaders for the following $(5,2)$ ternary code:

$$C = \{00000, 10201, 20102, 02120, 01210, 11111, 22222, 21012, 12021\}$$

   (Just kidding. How many elements would such a table have?) How many vectors of weight 1 are there in $\mathbb{Z}_3^5$? What is d_{min} and how many errors can this code correct?
5. Let $G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}$. This gives a $(4,2)$ ternary linear code C. Find H and d_{min}. Make a list of all coset leaders. Use syndrome decoding to decode $w = 2221$. Verify that your answer lies in C.
6. Estimate the computational cost of setting up a standard array, and compare this with the naive method of searching to decode each vector. Now estimate the computational cost of coset decoding, and compare with the naive search method. Finally, compare these costs with syndrome decoding. For simplicity, you may assume the codes are perfect.
7. The $(7,4)$ Hamming code[1] is the binary code C with generator matrix

$$G = \begin{bmatrix} I_4 \\ A \end{bmatrix}, \quad \text{where} \quad A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

   (i) Find a parity check matrix for C and use it to determine d_{min}. Is C perfect? MDS? What is its rate?

   (ii) Shannon proposed the following decoding algorithm. Define

$$a = 0001111, b = 0110011, c = 1010101.$$

   Given a received word w, calculate $x_1 = w \cdot a$, $x_2 = w \cdot b$ and $x_3 = w \cdot c$. Let $x = x_1x_2x_3$, viewed (!!) as a binary number (that is, $0 \le x \le 7$). If x is not zero, then w is the result of a single error in the xth bit. Show that this works for $w = 1110110$. Apply the usual syndrome decoding to verify the decoding.

   (iii) Prove that Hamming decoding (that is, the procedure outlined in this exercise) works.

5.5 Hamming codes

Hamming codes are an important class of perfect single-error correcting codes.

[1]This was the first "nontrivial" example of an error-correcting code, presented by Shannon in 1948.

Definition 5.16 Let $\ell \geq 2$ be an integer and let F be a finite field with q elements. Set

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

An $(n, n-\ell)$ q-ary code is a *Hamming code* if the columns of its parity check matrix are pairwise linearly independent.

Note that if the columns of H are pairwise linearly independent, then $d_{min} \geq 3$, by Theorem 5.11. Another way to say the columns are "pairwise linearly independent" is to say that no two columns are scalar multiples of one another — so this is an easy condition to check.

■ **Example 5.17** Take $\ell = 2$ and $q = 2$. Then $n = (2^2-1)/(2-1) = 3$ so this is a $(3,1)$ code. A parity check matrix H will be of size 2×3. None of its columns can be equal ($q = 2$), so up to ordering, the matrix must be

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

which yields the $(3,1)$ repetition code. ■

■ **Example 5.18** The $(7,4)$ code of Exercise 7 is a binary Hamming code, with $\ell = 3$, $q = 2$ and thus $n = (8-1)/(2-1) = 7$. ■

Proposition 5.19 Every nonzero vector of F^{n-k} occurs, up to scalar multiple, as a column of H, so the columns of the parity check matrix of a Hamming code form a *maximal* set of pairwise linearly independent vectors. In consequence, the minimal distance of a Hamming code is exactly 3.

Proof. The parity check matrix is size $(n-k) \times n = \ell \times n$. Thus its columns are vectors in F^ℓ. There are thus $q^\ell - 1$ possible nonzero columns. Two columns are linearly dependent if and only if they are scalar multiples of one another.

So define an equivalence relation $\sim$ on the set of nonzero vectors in F^ℓ by the rule:

$$v \sim w \quad \Leftrightarrow \quad \exists \lambda \in F \setminus \{0\} : v = \lambda w.$$

(Exercise: this is an equivalence relation.) Each equivalence class has exactly $q-1$ elements in it (the nonzero multiples of any element in it) and the equivalence classes must partition the set. Therefore the number of equivalence classes is

$$\frac{\text{number of nonzero vectors in } F^\ell}{\text{number of elements in each class}} = \frac{q^\ell - 1}{q-1}$$

which is n. We have to choose n columns of H and at most one column from each equivalence class, therefore, we must choose exactly one column from each class, and so our set of column vectors is maximal with respect to this property.

Moreover, since all nonzero vectors occur, up to scalar multiple, and $\ell \geq 2$, we have a set of three linearly dependent vectors: the sum of the first and second columns, for example, must be a multiple of another column. Therefore $d_{min}(C) = 3$. ■

From this proposition, we can quickly write down a parity check matrix for any Hamming code.

Theorem 5.20 Hamming codes are perfect single-error correcting codes.

Proof. Recall that a perfect code is one which meets the sphere-packing bound. We know, by the proposition, that any Hamming code has $d_{min} = 3$, so it correct $\mathbf{t} = 1$ error. We note that

$$q^k\left(\binom{n}{0} + (q-1)\binom{n}{1}\right) = q^k\left(1 + (q-1)\frac{q^\ell - 1}{q-1}\right) = q^k q^\ell = q^{k+\ell} = q^n$$

so the code is perfect. ■

Corollary 5.21 The only nontrivial linear perfect single error correcting codes are the Hamming codes.

Proof. Perfect and single error-correcting imply that (n,k) meets the sphere-packing bound with $\mathbf{t} = 1$. Well, as we saw in the proof above, this happens iff

$$1 + n(q-1) = q^{n-k} \quad \text{equivalently:} \quad n = \frac{q^{n-k} - 1}{q-1}.$$

Set $\ell = n - k \geq 0$. If $\ell = 0$, then we have $n = k$, so the code has $d_{min} = 1$ and corrects no errors. If $\ell = 1$, then the left side is 1 so $n = 1$ and $k = 0$, which corresponds to the zero vector space.[2] For $\ell \geq 2$, this is a Hamming code. ■

5.6 The search for perfect codes

So what are all the perfect error-correcting codes?

$\mathbf{t} = 1$: Hamming codes

$\mathbf{t} = 2$: Let's compute the Hamming bound in the binary case.

$$\begin{aligned}\binom{n}{0} + \binom{n}{1} + \binom{n}{2} &= 2^{n-k} \\ 1 + n + \frac{1}{2}n(n-1) &= 2^{n-k} \\ n^2 + n + 2 &= 2^{n-k+1}\end{aligned}$$

For which values of n is this a power of 2? We check a few values and see $n = 1,2,5$ as possible solutions. Now $n = 1,2$ are too small (they certainly can't contain a code with $d_{min} = 5$) but for $n = 5$ we have the $(5,1)$ repetition code

$$C = \{00000, 11111\}$$

which is 2-error correcting. Moreover, $2^{n-k+1} = 2^5 = 32$, so it is perfect.

[2]This is an error-correcting code, but since there's only one codeword, it is called trivial.

Beyond this, what other values of n work? You're looking for integer solutions to $n^2+n+2=2^m$. These are the kinds of questions brilliant mathematicians like Srinivasa Ramanujan explored and he conjectured the answer in 1913 (proven later in 1948), though in a slightly different form (now called the Ramanujan–Nagell equation; its Wikipedia page has a nice derivation of the equivance to our equation). And in fact, the only other integer solution to this equation is $n=90$, with $n-k+1=13$. But later it was shown that no perfect binary $(90,78)$ code exists (even if you allow nonlinear codes!).

Perfect codes are very hard to find, and variants on this question are a subject of ongoing research. The story is settled for linear codes, though, with the following theorem that has been known since 1975 [LIN75].

Theorem 5.22 The only perfect linear error-correcting codes are:

1. binary $(2n+1,1)$ repetition codes;
2. Hamming codes;
3. the Golay ternary $(11,6)$ code with $d_{min}=5$;
4. the Golay binary $(23,12)$ code with $d_{min}=7$.

These mysterious Golay codes are examples of cyclic codes — reason enough to tackle them next.

5.7 Another desirable structure: cyclic codes

Thus far, we have established many properties of linear codes, and can say many things about a code we are given. But besides Hamming codes, we have seen few constructions that yield entire classes of codes from which we could choose for a given application; in particular, we need ways to choose good codes of high dimension, and with high error-correcting capabilities.

A nice way to construct codes is to look for classes of codes with additional structure, and then to relate this to some algebraic structures.

Definition 5.23 Given a vector $c=c_1\cdots c_n$, we define its *cyclic shift* to be the vector

$$c^1=c_nc_1c_2\cdots c_{n-1}.$$

For any integer $r\geq 1$, we recursively define $c^r=(c^{r-1})^1$. For example $c^2=(c^1)^1$, which is a shift by 2.

Note that $c^1=\sigma(c)$ where $\sigma=(1\,2\,3\cdots n)$ is the full cyclic permutation, since

$$\sigma(\sum_i c_ie_i)=\sum_i c_i\sigma(e_i)=\sum_i c_ie_{i+1}$$

where we understand $i+1$ to mean $i+1$ mod n; thus the coefficient of $e_1=e_{n+1}$ is c_n and the coefficient of any other e_k is c_{k-1}.

Definition 5.24 A code C is *cyclic* if for all $c\in C$, we also have that $c^1\in C$, that is, the first cyclic shift of c is again in C. Note that this implies that all cyclic shifts of c lie in C.

■ **Example 5.25** $C=\{000,011,101,110\}$ is a cyclic code. ■

■ **Example 5.26** Let C be the $(7,4)$ Hamming code with generator matrix

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

Is this cyclic?

Say we begin with $c = 1000110$. Then we have

- $c^1 = 0100011 \in C$, good
- $c^2 = 1010001$ is the sum of the first and third columns of G, so in C
- $c^3 = 1101000$ is the sum of columns 1, 2 and 4, so in C.

We could keep going, but to check each vector in C (how many are there?) would be a pain. Instead, we can see that these four vectors are linearly independent, hence form a basis for C. So for example

$$c^4 = c + c^1 + c^2 \in C$$

and furthermore, once this relation is known, we can write down several more, for free:

$$c^5 = c^1 + c^2 + c^3$$
$$c^6 = c^2 + c^3 + c^4 = c + c^1 + c^3$$
$$c^7 = c^1 + c^2 + c^4 = c$$

where this last one is what we expected.

Moreover, we claim that we have done enough work to prove this code is cyclic. Namely: suppose $c' \in C$. Then $c' \in \operatorname{span}\{c, c^1, c^2, c^3\}$ since these form a basis. Thus

$$(c')^1 \in \operatorname{span}\{c^1, c^2, c^3, c^4\} = \operatorname{span}\{c^1, c^2, c^3, c + c^1 + c^2\} = \operatorname{span}\{c, c^1, c^2, c^3\} = C,$$

as required. ■

Let's generalize the technique we used in this last example.

Lemma 5.27 Suppose C has a basis B such that for each $c \in B$, we have $c^1 \in C$. Then C is cyclic. In this case, every basis has this property.

Proof. We note that for any scalars α, β and any vectors u, v,

$$(\alpha u + \beta v)^1 = \alpha u^1 + \beta v^1;$$

in other words, the cyclic shift is a linear transformation on F^n, for any n. Furthermore, if $u^1 = 0$ then $u = 0$, so it is injective, hence an isomorphism. Therefore, the cyclic shift takes a basis for C to a basis for C^1, the image of C under the cyclic shift.

So if $B^1 \subseteq C$, it follows that $C^1 \subseteq C$, and so they are equal.

A less theoretical approach to this argument would be: any vector in C is a linear combination of the basis vectors, say $c = \sum a_i u_i$, with $u_i \in B$. By linearity, $c^1 = \sum a_i u_i^1 \in C$.

The last statement follows from the definition of a cyclic code: the cyclic shift of *every* vector of C lies in C. ■

R Like the property of being systematic, the property of being cyclic is NOT an invariant of the equivalence class. In other words, a cyclic code and a non-cyclic code could be equivalent; the choice of ordering (permutation) of the coordinates is important.

The theorem tells us that being cyclic is something we can see independent of our choice of basis (unlike systematic codes, where we had to choose a correct basis to prove that the code was systematic).

Proposition 5.28 If C is cyclic, so is $C^\perp$.

Proof. Suppose $h \in C^\perp$; we need to prove that $h^1 \in C^\perp$. So let $c \in C$. We need to show that $h^1 \cdot c = 0$. We calculate

$$\begin{aligned}
h^1 \cdot c &= (h_n, h_1, \cdots, h_{n-1}) \cdot (c_1, c_2, \cdots, c_n) \\
&= h_n c_1 + h_1 c_2 + \cdots + h_{n-1} c_n \\
&= h_1 c_2 + \cdots + h_{n-1} c_n + h_n c_1 \quad \text{(first term moved to end)} \\
&= (h_1, h_2, \cdots, h_n) \cdot (c_2, c_3, \cdots, c_n, c_1) \\
&= h \cdot c^{n-1}
\end{aligned}$$

and since C is cyclic, we have $c^{n-1} \in C$, so this last is zero since $h \in C^\perp$. ■

Can we construct a cyclic code? Yes! Choose one or more linearly independent vectors, then compute their cyclic shifts. If any of them is linearly independent from the vectors you started with, add them to the set, and repeat.

■ **Example 5.29** What is the cyclic code in $\mathbb{Z}_2^5$ of minimal dimension containing $c = 10100$? We compute

$$c^1 = 01010, c^2 = 00101, c^3 = 10010, c^4 = 01001, c^5 = c.$$

So: the minimal cyclic code C containing c is the span of these vectors. To find a basis, we row reduce:

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

whence we deduce that $\dim(C) = 4$ and moreover that a basis is

$$\{10001, 01001, 00101, 00011\}.$$

Ah: this is a parity check code! That was a lot of work to produce a not super-interesting result. ■

Cyclic codes end up being central to the theory of linear error-correcting codes for a few reasons. For one, they admit a very efficient encoding and decoding algorithms using “shift registers" (Chapter 7.8) which greatly improves on syndrome decoding. For another, we will see that it is possible to *design* a cyclic code with a given minimum distance — a huge advantage over the few random examples we have seen to date.

But to access all this structure: we need some ring theory.

5.8 Exercises

1. Prove that any code that is equivalent to a Hamming code is again a Hamming code.
2. Prove that any two binary Hamming codes are equivalent.
3. Prove than any two q-ary Hamming codes are scaled-equivalent (see §4.7, Exercise 13).
4. Prove that the minimum distance of any Hamming code is exactly 3. (Note that the definition of the parity check matrix gave us $d_{min} \geq 3$; and the sphere-packing bound further shows $d_{min} \leq 4$.)
5. A code is called an *equidistant code* if for all $x, y \in C$, such that $x \neq y$, $d(x,y) = d_{min}$. These codes make the best use of space, in some sense, in that each pairwise probability of error is equal. The dual of a Hamming code is called a *simplex code*. Prove that every simplex code is equidistant.
6. Prove that the dual of a maximum distance separable code is again maximum distance separable. Recall that this means it meets the Singleton bound.
7. A code is self-dual if $C^{\perp} = C$. Prove that an (n,k) code with generator matrix G is self-dual if and only if $2k = n$ and $G^T G = 0$. Express this as conditions on the columns of G, and then give a generator matrix of a binary $(10,5)$ self-dual code.
8. Show that the ternary $(12,6)$ Golay code with generator matrix
$$G = \begin{bmatrix} I_6 \\ A \end{bmatrix} \quad \text{with} \quad A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{bmatrix}$$
has $d_{min} = 6$.
9. Prove that every binary parity check code is cyclic.
10. Show explicitly that the dual of the $(7,4)$ Hamming code (a simplex code) of Example 5.26 is cyclic.
11. Find a $(7,4)$ Hamming code that is NOT cyclic, thereby proving the assertion that being cyclic is not invariant under equivalence.
12. Lemma 5.27 does not address a related question: in Example 5.26, we could choose a basis such that all elements were cyclic shifts of the first vector. Is this always true? (We will use ring theory to answer this question in the next chapter.)
13. Can you always choose a basis B for a cyclic code with the property that B itself is closed under cyclic shifts?

II Linear codes from polynomial rings

6. Rings

Ring theory is a vast subject, encompassing a huge variety of structures; virtually any reasonable-looking set with two operations will turn out to be a ring. We have a whole course devoted to the exciting diversity of rings (MAT3143).

For this course, we need rings for two major reasons. For one, cyclic codes are made from certain rings. For another, we have already seen that *fields*, especially finite fields, are of critical interest. To classify them (Chapter 8), we'll need more ring theory (and then we'll use that classification to construct the best linear codes).

6.1 Definition of a ring

We start with the definition.

Definition 6.1 A *ring* is a set R with 2 operations, addition and multiplication, such that

1. $(R,+)$ is an abelian group;
2. R is closed under multiplication $\cdot$, and $\cdot$ is associative;
3. multiplication distributes over addition on the left and the right, that is, for all $a,b,c \in R$, $a(b+c) = ab+ac$ and $(a+b)c = ac+bc$.

If R contains an identity 1 for multiplication, then we call it a *unital ring*; if multiplication is commutative, then we call R a *commutative ring*.

Some quick notes:

- The set of elements of R that have a multiplicative inverse is denoted $R^\times$ and its elements are sometimes called the *units of R*.
- A commutative unital ring in which every nonzero element of R admits a multiplicative inverse (where $R^\times = R \setminus \{0\}$) is a *field* (see Definition 2.6).

For us, a *ring* will be a commutative unital ring.

■ **Example 6.2** $\mathbb{Z}$ is a ring, but it is not a field. ■

■ **Example 6.3** $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ is a ring for any $n \geq 2$. The set of its units is

$$\mathbb{Z}_n^\times = \{x \in \mathbb{Z}_n \mid \gcd(x,n) = 1\}$$

so $\mathbb{Z}/n\mathbb{Z}$ is a field only if n is prime. ■

■ **Example 6.4** Let F be a field. Then the polynomial ring

$$F[x] := \{a_0 + a_1 x + \cdots + a_N x^N \mid a_i \in F, N \geq 0\}$$

is a ring, where the operations are just addition of polynomials, and multiplication of polynomials, in the usual way. Note that $F[x]$ can also be thought of as a vector space over F if we only think about the multiplication by scalars (polynomials of degree 0); in this case it is an infinite dimensional vector space over F. ■

■ **Example 6.5** The polynomial ring $\mathbb{Z}[x]$, where the coefficients take values in the integers, is also a ring. ■

■ **Example 6.6** Dropping the condition that multiplication is commutative yields the class of non-commutative rings; examples of non-commutative rings include the ring of 2×2 matrices with entries in $\mathbb{Z}$, or in a field. Dropping the condition that multiplication has an identity element yields the class of nonunital rings; examples include the ring of compactly-supported functions on $\mathbb{R}$. ■

For the purposes of coding theory, and of (more complicated) finite fields, the important rings to understand are polynomial rings over finite fields. Let us explore some of their unique properties, and see how they are quite similar in structure, in some ways, to the ring of integers. For more detail on the material in this chapter, see [Nic12, Chapter 3.1].

6.2 Polynomial rings over fields

You're familiar with $\mathbb{R}[x]$; our objective in this section is to go through the properties we know to be true of $\mathbb{R}[x]$, and prove that they hold for $F[x]$, for any field F. For more detail on the material in this section, see [Nic12, Chapters 4.1 and 4.3].

Recall from Example 6.4 that if F is a field then the polynomial ring (in one variable) over F is

$$F[x] := \{a_0 + a_1 x + \cdots + a_N x^N \mid a_i \in F, N \geq 0\},$$

equipped with the usual addition and multiplication of polynomials. If $f \in F[x]$, either $f = 0$ or else there is a unique $N \geq 0$ and unique coefficients $a_0, \ldots, a_N \in F$ with $a_N \neq 0$ such that

$$f = a_0 + a_1 x + \cdots + a_N x^N.$$

In this case, a_0 is called the *constant term*, and $a_N x^N$ is called the *leading term*. A polynomial is *monic* if $a_N = 1$. The number N is called the *degree of* f. If $f = 0$ then we sometimes say $\deg(0) = -\infty$, but it is also common to include f among the *constant polynomials*, which are the polynomials of the form $f(x) = a_0$, for any $a_0 \in F$ (but which are sometimes just called the polynomials of degree 0).

Let's begin with the key property that $F[x]$ has in common with $\mathbb{Z}$ (making it in particular a *Euclidean ring*).

Lemma 6.7 — Division algorithm. Let F be a field. Let $a,b \in F[x]$, with $b \neq 0$. Then there exist unique polynomials $q,r \in F[x]$ such that

$$a = qb + r$$

where either $r = 0$ or $0 \leq \deg(r) < \deg(b)$.

This result is a familiar one, at least over $F = \mathbb{R}$; as we go through the proof (which is quite a clever one), we should think about what properties we require of F for it to hold. For example, this theorem is not true if we replace F by $\mathbb{Z}$.

Proof. Let $a,b \in F[x]$, with $b \neq 0$. If $a = 0$, then we may take $q = r = 0$; we leave it as an exercise to show that this is the only solution.

Now suppose $a \neq 0$ and define

$$S = \{r \in F[x] \mid r = a - qb, \quad \text{for some } q \in F[x]\}.$$

If $0 \in S$, then $a = qb$ for some q, so we could take $r = 0$; this is the only solution in this case since if we had $a = q'b + r'$, with $r' \neq 0$, we'd have $(q - q')b = r'$. If $q \neq q'$, the left hand side has degree $\geq \deg(b)$ while the right hand side has degree $< \deg(b)$; a contradiction. So $q' = q$ and thus $r' = 0$ and thus the solution is unique.

Otherwise, let $r \in S$ be an element of minimal degree, and write $r = a - bq$ for some $q \in F[x]$. To show that this is a solution, we need to prove that $\deg(r) < \deg(b)$. For suppose not: then write

$$\begin{aligned} r &= r_0 + \cdots + r_n x^n, \quad r_n \neq 0 \\ b &= b_0 + \cdots + b_m x^m, \quad b_m \neq 0 \end{aligned}$$

with $m \leq n$. Set

$$r' = r - \left(\frac{r_n}{b_m} x^{n-m}\right) b.$$

Then this is clearly an element of S, and by construction either $r' = 0$ or $\deg(r') < \deg(r)$, contradicting the choice of r. Hence $\deg(r) < \deg(b)$.

We leave the proof of uniqueness as an exercise. ■

We see directly that this proof does not apply to $\mathbb{Z}[x]$, for example; in fact, we can use the proof to generate counterexamples (exercise).

The division algorithm has many wonderful consequences. As in $\mathbb{Z}$, you can use this algorithm to form the Euclidean algorithm, which computes the gcd of any two elements of $F[x]$. As a consequence, the gcd exists [Nic12, §4.2, Theorem 10], which implies that every irreducible is prime [Nic12, §4.2, Theorem 11], which implies uniqueness of factorization [Nic12, §4.2, Theorem 12]. Also, since roots of h correspond to linear factors of h (exercise), we may deduce that if $h \in F[x]$ has degree n, then h has at most n roots, counting with multiplicity.

We will accept these as facts about $F[x]$, and instead focus on how to calculate the greatest common divisor, or how to find out if a polynomial is irreducible.

6.3 Greatest common divisor

For a refresher on the gcd of integers, please see Appendix A.1.

To do this in polynomial rings, there is a subtle point. Over the integers, we think of the gcd as producing a unique answer, but for example $\gcd(6,-9) = \pm 3 = \gcd(6,9)$. The answer is unique only up to a *unit*, that is, an invertible element of the ring. Over $\mathbb{Z}$ the only units are ± 1; in $F[x]$, the units are exactly the degree zero polynomials $a \in F^{\times}$, which we identify with the set of nonzero scalars. Hence we might get an answer like

$$\gcd(2x+4, 3x+6) \text{ is both } 2x+4 = 2(x+2) \text{ and } 3x+6 = 3(x+2)$$

which is a bit perturbing. So we make the convention to scale the answer so that it is a monic polynomial; this is akin to choosing the positive sign when we work over $\mathbb{Z}$.

■ **Example 6.8** A quick reminder: to calculate the gcd of 35 and 10, we compute

$$\begin{aligned} 35 &= 3 \times 10 + 5 \\ 10 &= 2 \times 5 + 0 \end{aligned}$$

so $\gcd(35,10) = 5$. ■

■ **Example 6.9** Calculate, over $F = \mathbb{Z}_5$, $\gcd(2x^3+4x^2+3, 4x^2+3x+3)$.

Set $a = 2x^3+4x^2+3$ and $b = 4x^2+3x+3$. Do long division; we use the property that $4x^2 \times 3x = 2x^3$ over $\mathbb{Z}_5$, for example, to compute the answer. We get

$$a = (3x)b + (x+3).$$

Now do long division with $x+3$ and b; we find it divides evenly:

$$b = (4x+1)(x+3) + 0$$

so the gcd is $x+3$. ■

×	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

■ **Example 6.10** Now calculate $\gcd(2x^3+4x^2+3, 4x^2+3x+3)$ over $\mathbb{Z}_7$.

This takes many more steps. (Exercise) Using the notation a,b as in the preceding example, we obtain the sequence of division algorithm steps:

$$\begin{aligned} a &= (4x+5)b + (x+2) \\ b &= (4x+2)(x+2) + 6 \\ x+2 &= (6x+5)6 + 0 \end{aligned}$$

so that after scaling, $\gcd(a,b) = 1$; so over $\mathbb{Z}_7$ these polynomials are relatively prime. ■

×	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

An important consequence of the division algorithm is that you can trace back through the calculation of $\gcd(a,b) = d$ to write $d = sa + tb$ for some $s,t \in F[x]$.

■ **Example 6.11** From Example 6.8, the first equation yields $5 = 1 \times 35 - 3 \times 10$ so $s = 1$ and $t = -3$.

From Example 6.9, we deduce $x+3 = a - (3x)b$ so $s = 1$ and $t = -3x$.

From Example 6.10 we have to do a bit more work, starting from the second-last equation and working our way upwards until everything is written in terms of a and b:

$$\begin{aligned}6 &= b-(4x+2)(x+2) = b-(4x+2)\,(a-(4x+5)b)\\ &= -(4x+2)a + (1+(4x+2)(4x+5))b = -(4x+2)a+(2x^2+4)b.\end{aligned}$$

Now note that $6 = -1$ to conclude that $(4x+2)a - (2x^2+4)b = 1$, as desired. ■

6.4 Irreducible polynomials

A nonconstant polynomial h is *irreducible* if it cannot be factored as $h = fg$ with $1 \le \deg(f), \deg(g) < \deg(h)$.

Notice that the irreducibility of a given polynomial depends on the field over which we are working. For example,

$$x^2+x+1$$

is irreducible over $\mathbb{Z}_2$ (since if it factored, the factors would be linear, and hence correspond to roots, of which there are none), but factors over $\mathbb{Z}_3$ as

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

■ **Example 6.12** Find all the irreducible quadratic polynomials over $\mathbb{Z}_2$.

We have a few options, but the simplest is to enumerate all the polynomials of degree two over $\mathbb{Z}_2$ and work out which ones are irreducible. We are greatly helped in this exercise by the fact that any factorization would have to be into linear factors, which means the polynomial would have to have a root (in the binary field).

The quadratics are: x^2, x^2+1, x^2+x, x^2+x+1. The first three are obviously not irreducible, and the last is irreducible as we noted above. ■

■ **Example 6.13** Find all irreducible quartics over the binary field.

Again, let's begin by enumerating all the polynomials of degree 4 over $\mathbb{Z}_2$. Well, we can omit any which do not have a constant term, because they would have x as a linear factor. We can also omit any which have an even number of terms, because they obviously admit $x = 1$ as a root. This leaves just 4 to write down:

$$x^4+x+1, x^4+x^2+1, x^4+x^3+1, x^4+x^3+x^2+x+1$$

None of these has any roots, but that is not enough to deduce irreducibility. A polynomial is irreducible iff it cannot be factored; a polynomial of degree 4 which factors will either have a root, or will have two factors of degree 2.

But now we use our previous exercise. If a quartic has no roots but factors, then it must factor as a product of two irreducible quadratics. There is only one such quadratic; and we calculate:

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

We conclude that the remaining 3 quartics above are irreducible. ■

6.5 Exercises

1. Suppose a ring R contains a field (with respect to the same operations). Show that R is therefore a vector space over F. To which over the following pairs (R,F) does this statement apply?
   (a) $R = F[x]$, F (identified with the constant polynomials)
   (b) $R = \mathbb{Z}$, $F = \mathbb{Z}_p$
   (c) $R = \mathbb{Z}/p^2\mathbb{Z}$, $F = \mathbb{Z}_p$
   (d) R is the ring of all functions from a set X to F; F is identified with the set of constant functions

   Hint: the key trick here is to decide if F is a subring, that is, its structure is with respect to exactly the same operations as R.
2. Show that if F is a field, then for $a,b \in F[x]$, $ab = 0$ iff at least one of a or b is zero.
3. Show that if F is a field, and $a,b \in F[x]$ with $a,b \neq 0$, then $\deg(ab) = \deg(a)+\deg(b)$.
4. Suppose $a,b \in F[x]$. Argue that if $a = 0$ and $b \neq 0$, then the only solution to $a = qb+r$ with $q,r \in F[x]$ and either $r = 0$ or $\deg(r) < \deg(b)$, is the trivial solution $q = r = 0$.
5. Complete the proof of the division algorithm for $F[x]$ by showing the answer is unique.
6. Show that there exist $a,b \in \mathbb{Z}[x]$, $a,b \neq 0$, for which there are no q,r as in the division algorithm theorem.
7. Find $\gcd(3x^4+x^3+3x^2+4x+4, x^4+4)$ in $\mathbb{Z}_5[x]$.
8. Find $\gcd(3x^4+x^3+3x^2+4x+4, x^4+4)$ in $\mathbb{Z}_7[x]$.
9. Find all irreducible binary cubic polynomials.
10. Find all irreducible ternary monic quadratic polynomials.
11. Find all irreducible binary quintic polynomials.
12. How many monic polynomials of degree n are there in $F[x]$, where F is a finite field with q elements?

6.6 Ideals

Now that we have the abstract concept of a ring (and our particular example of a polynomial ring over a field), we proceed as we usually do with a new definition of an algebraic object: we look for interesting subobjects (and interesting morphisms). For more detail on the general theory of ideals, see [Nic12, Chapter 3.3].

Let's begin with the example of $\mathbb{Z}$. The interesting subobjects are sets like

$$n\mathbb{Z} = \{na \mid a \in \mathbb{Z}\}.$$

They are additive subgroups, and also closed under multiplication, but they don't contain 1 (unless $n = \pm 1$). There's something even more interesting about them: this set isn't *just* closed under multiplication of two elements within the set (although it is) but actually closed under multiplication by *any* element of the entire ring. That is, if we take any integer, and multiply it by an element of $n\mathbb{Z}$, we end up back in $n\mathbb{Z}$. This wonderful property is so nice we call it an ideal.

Definition 6.14 Let R be a ring. A subset I of R is called a (left) *ideal* of R if

- I is an additive subgroup; and
- for all $a \in I$, for all $r \in R$, $ra \in I$.

It is called a right ideal if for all $a \in I$, for all $r \in R$, $ar \in I$. If R is commutative, then left ideals are the same as right ideals, and we just call I an ideal. We write $I \lhd R$.

R You could make the parallel: subgroups are to subrings as normal subgroups are to ideals. Technically, however, note that $n\mathbb{Z}$ doesn't contain 1, so in the category of unital rings, this is not a subring. This is part of the reason we're not going to bother with subrings at all, but just focus on ideals.

■ **Example 6.15** The set of even integers, $2\mathbb{Z}$, is an ideal of $\mathbb{Z}$. ■

■ **Example 6.16** Consider $R = \mathbb{Z}[x]$. The set

$$I = \{2a_0 + a_1x + \cdots + a_Nx^N \mid N \geq 0, a_i \in \mathbb{Z}\}$$

is an additive subgroup; and if you multiply two polynomials, the constant term of the product is just the product of the constant terms, so you deduce that if one polynomial has even constant term, so does any product. Therefore I is an ideal of R.

If instead we'd taken $I' = \{a_0 + 2a_1x + a_2x^2 + \cdots + a_Nx^N \mid N \geq 0, a_i \in \mathbb{Z}\}$ then this would not be an ideal, since for example $(1+2x)(3x) = 3x + 6x^2 \notin I$. ■

■ **Example 6.17** Note that $\mathbb{Z}[x] \subset \mathbb{R}[x]$ is an additive subgroup, and a subring, but it's not an ideal, since $\frac{1}{2}p \notin \mathbb{Z}[x]$ for any $p \in \mathbb{Z}[x] \setminus \{0\}$. ■

Definition 6.18 Let $p \in R$, a commutative ring. Then the *ideal generated by p* is the set

$$\langle p \rangle := \{pq \mid q \in R\}.$$

When an ideal is generated by a single element, we call it a *principal ideal*.

We can quickly check that it is in fact an ideal: if $pq, pq' \in \langle p \rangle$ (meaning: $q, q' \in R$) then $(pq) + (pq') = p(q+q')$, which is in $\langle p \rangle$ since $q + q' \in R$. So it's closed under addition. If $r \in R$, then $r(pq) = p(rq) \in \langle p \rangle$, so it is closed under multiplication by any element of R. Thus it's an ideal.

■ **Example 6.19** The ideal $n\mathbb{Z}$ of $\mathbb{Z}$ is a principal ideal, generated by the element n. ■

■ **Example 6.20** Let $F = \mathbb{Z}_3$, $R = F[x]$ and $p = x - 1$. Then the ideal generated by p is

$$\langle p \rangle = \{(x-1)q \mid q \in F[x]\}$$

and this is exactly the set of all polynomials in $F[x]$ that have 1 as a root. ■

R We sometimes write pR or Rp in place of $\langle p \rangle$, but note that this is not like a coset in a multiplicative group (since R is not a multiplicative group) — it's the subset of all multiples of p.

Where the ring theory for polynomial rings over fields really takes off is the following theorem.

Theorem 6.21 Let R be $\mathbb{Z}$, or $F[x]$ for some field F. Then every ideal is principal.

Proof. Suppose $a, b \in I$. Then since R has a Euclidean algorithm, we can compute

$$g = \gcd(a,b) = ra + sb$$

for some $r, s \in R$. But since $a, b \in I$, so are ra, sb; and since I is closed under addition, $ra + sb = g \in I$. Conclusion: the gcd of any two elements of my ideal is again in my ideal.

So now suppose I is nonzero, and choose a nonzero element $a \in I$ of least
- absolute value, if $R = \mathbb{Z}$;
- degree, if $R = F[x]$.

Now let $b \in I$; we'll show a divides b. Since $a, b \in I$, their gcd $g \in I$. By minimality of a, we know that the absolute value or degree of g equals that of a. But g divides a. This means there exists some $r \in R$ such that $rg = a$. But then r is an element of
- absolute value 1 (case $R = \mathbb{Z}$), since $|a| = |rg| = |r| \times |g|$; or
- degree 0 (case $R = F[x]$), since $\deg(a) = \deg(rg) = \deg(r) + \deg(g)$.

In either case, r is a unit. But g divides b so therefore $r^{-1}a$ divides b; but if $s(r^{-1}a) = b$, then $(sr^{-1})a = b$, so a divides b.

Thus I is a principal ideal, generated by a. ■

So we get all ideals of $F[x]$ by running through all possible choices of polynomials p and creating $\langle p \rangle$. Let's get a sense of how big these ideals will be.

Lemma 6.22 Suppose $I \lhd R$, where R is a ring which contains 1. Recall that $R^\times$ is the set of invertible elements R. Then
- if $1 \in I$ then $I = R$;
- if $I \cap R^\times \neq \emptyset$, then $I = R$.

So $\langle u \rangle = R$ for any $u \in R^\times$.

The proof is an exercise.

In particular, for $R = F[x]$, where $R^\times = F^\times$, we infer that if $a_0 \in F$ is a constant polynomial then

$$\langle a_0 \rangle = \begin{cases} F[x] & \text{if } a_0 \neq 0; \\ \{0\} & \text{if } a_0 = 0. \end{cases}$$

So that's not very interesting.

Lemma 6.23 Let $a,b \in R$. Then

$$\langle a\rangle \subseteq \langle b\rangle \iff \exists q \in R, \quad a = bq \iff b \text{ divides } a.$$

That is, the ideal generated by a contains the ideal generated by b if and only if $b|a$.

Note how this says that "smaller elements give bigger ideals".

Proof. Suppose b divides a, that is, there is some $q \in R$ for which $bq = a$. Then $a \in \langle b\rangle$, by definition. Thus all multiples of a will also lie in $\langle b\rangle$, which means $\langle a\rangle \subseteq \langle b\rangle$. Conversely, if $\langle a\rangle \subseteq \langle b\rangle$, then in particular, the element a of $\langle a\rangle$ lies in $\langle b\rangle = \{qb \mid q \in R\}$. So there exists $q \in R$ such that $a = qb$. ■

6.7 Quotient rings

We stated earlier that ideals are the ring analogues of normal subgroups. Here, we make this precise, by showing that they are the right objects for defining quotients which are again rings. For more details on general factor rings, see [Nic12, Chapter 3.3]. For the particular case of polynomial rings, see [Nic12, Chapter 4.3].

Definition 6.24 Let R be a ring and I an ideal. Then the *quotient ring* or *factor ring*, denoted R/I, is the set of additive cosets of I in R:

$$R/I = \{a+I \mid a \in R\}$$

with addition and multiplication defined by the rules

- $(a+I)+(b+I) = (a+b)+I$, for all $a,b \in R$
- $(a+I)(b+I) = ab+I$, for all $a,b \in R$

Of course we must first show that this is indeed a ring.

Proof that the quotient ring is a ring. Since I is a subgroup of the abelian group R, it is a normal subgroup, so R/I is already automatically an abelian group, with the given addition. For the multiplication, we see that the only issue is whether or not it is well-defined; after that, associativity and commutativity and existence of 1 follow from R easily.

So what needs to be checked? The formula produces an answer in R/I, that's no problem. But recall that the coset representatives are not unique. We need to show that if $a+I = a'+I$ and $b+I = b'+I$ that $ab+I = a'b'+I$, that is, that our definition is independent of the choice of representative.

Well, if $a+I = a'+I$ then there is some $c \in I$ such that $a = a'+c$; and if $b+I = b'+I$ then there is some $d \in I$ such that $b = b'+d$. Then

$$ab = (a'+c)(b'+d) = a'b'+cb'+da'+cd.$$

But $cb', da', cd \in I$ since $c,d \in I$, and I is an ideal. So ab and $a'b'$ differ by an element of I, so $ab+I = a'b'+I$, as required. ■

Note that we absolutely needed the "closure under multiplication by arbitrary elements of the ring" condition on an ideal to ensure that multiplication is well-defined.

■ **Example 6.25** For any n, $\mathbb{Z}/n\mathbb{Z}$ is the ring we denote $\mathbb{Z}_n$: the coset $a+n\mathbb{Z}$ is the set of all integers that are congruent to a modulo n. We usually choose our favourite set of coset representatives $\{0,1,\cdots,n-1\}$, do our usual operations, and then take the remainder upon division by n to convert our answer back to our favourite representatives. ■

That is, in $\mathbb{Z}/n\mathbb{Z}$ we have a shortcut that makes it seamless to work with cosets: instead of writing

$$(5+8\mathbb{Z})+(7+8\mathbb{Z}) = 12+8\mathbb{Z} = 4+8\mathbb{Z}$$

we instead identify each coset by a favourite small representative and do the addition modulo n, as in

$$5+7 = 12 \equiv 4 \mod 8.$$

This is much easier to look at and think about!

Our next goal is to find the same kind of shortcut for quotient rings $F[x]/I$.

6.8 Writing elements of a quotient of the polynomial ring explicitly

Lemma 6.26 Let $a \in F[x]$ be any polynomial of positive degree, and set $I = \langle a \rangle$. For each coset $f+I$, there exists a unique representative f of minimal degree, and in fact it has degree strictly less than the degree of a.

Proof. Suppose f is a representative of its coset of minimal degree. If $\deg(f) \geq \deg(a)$, then we can perform long division to obtain

$$f = qa + r$$

where $\deg(r) < \deg(a)$. Furthermore, since $f-r = qa \in I$, it follows that $f+I = r+I$, so r is another coset representative, of smaller degree. This is a contradiction (even if $r=0$, because this would imply $f \in I$, so we can and do take $f=0$ as the representative). Hence $\deg(f) < \deg(a)$.

Next, suppose $f+I = g+I$ and both f and g have degree less than $\deg(a)$. Then $f-g \in I$, so a divides $f-g$. This means there exists a $q \in F[x]$ such that $aq = f-g$. But the degree of the left side is $\deg(a)+\deg(q)$, and the degree of the right side is at most the maximum of $\deg(f)$ or $\deg(g)$, both of which are less than $\deg(a)$. The only solution is $q=0$ and so $f=g$, whence unicity. ■

We deduce that if $\deg(a) = m$, then a set of representatives of the cosets in $F[x]/\langle a(x) \rangle$ is

$$\tilde{R} = \{c_0 + c_1 x + \cdots + c_{m-1}x^{m-1} \mid c_i \in F\}.$$

Given $f, g \in \tilde{R}$, we have that $(f+I)+(g+I) = (f+g)+I$, and since addition does not increase the degree of a polynomial, the sum $f+g$ is again in $\tilde{R}$.

On the other hand, when we multiply, $(f+I)(g+I) = fg+I$, and fg may no longer be a minimal representative. In this case, we proceed as in the proof of the lemma and perform the division algorithm:

$$fg = qa + r$$

so $fg+I=r+I$ and $r\in\tilde{R}$. So we define, in the set $\tilde{R}$, that

$$fg\equiv r \mod I.$$

Since R/I is a ring, and all we have done is renamed the elements of R/I, it follows that $\tilde{R}$ is also a ring. In future, we may abuse notation and just call it R/I.

Lemma 6.27 Let $\deg(a)=m$. Then

$$F[x]/\langle a\rangle=\{c_0+c_1x+\cdots+c_{m-1}x^{m-1}\mid c_i\in F\},$$

with usual addition, and with multiplication mod $\langle a\rangle$.

■ **Example 6.28** $F=\mathbb{Z}_5$, $R=F[x]$, $a=x^2+2x+2$, $I=\langle a\rangle$. Then we have

$$R/I=\{b_0+b_1x\mid b_0,b_1\in F\}.$$

For example, $(x+1)+(x+4)=2x\in R/I$, and $(x+1)(x+4)=x^2+4\notin R/I$. Subtracting a yields $x^2+4\equiv(x^2+4)-(x^2+2x+2)=3x+2\in R/I$. So we might write

$$(x+1)(x+4)\equiv 3x+2 \mod I$$

to avoid confusion.

This does suggest another way of approaching the long division. Since

$$a\equiv 0 \mod I$$

we have

$$x^2+2x+2\equiv 0 \mod I$$

which simplifies to

$$x^2\equiv 3x+3 \mod I.$$

So we can replace every occurrence of x^2 with a lower-degree expression. For example, $x^2+4\equiv(3x+3)+4=3x+2 \mod I$. ■

Now let's proceed to an important example, the one that brings us back to cyclic codes.

■ **Example 6.29** Let $F=\mathbb{Z}_p$. Then the polynomial

$$a=x^n-1$$

is NEVER irreducible; it always has a root at $x=1$. By the Lemma, we may write

$$F[x]/\langle x^n-1\rangle=\{c_0+c_1+\cdots+c_{n-1}x^{n-1}\mid c_i\in F\}$$

where multiplication obeys

$$x^n\equiv 1 \mod a.$$

■

6.9 Exercises

1. Prove Lemma 6.22. Suppose R is a field. What are the ideals of R?
2. Define a *ring homomorphism* $\varphi\colon R \to S$ between rings as a map which satisfies, for all $a,b \in R$, $\varphi(a+b) = \varphi(a)+\varphi(b))$, $\varphi(ab) = \varphi(a)\varphi(b)$ and $\varphi(1) = 1$.
   (i) Show that the inclusion map of $\mathbb{Z}$ into $\mathbb{R}$ is a ring homomorphism.
   (ii) Show that the multiplication by n map from $\mathbb{Z}$ to $\mathbb{Z}$ is not a ring homomorphism.
   (iii) Show that for any $a \in F$, the *evaluation map* $\varphi_a\colon F[x] \to F$ given by $p \mapsto p(a)$ is a ring homomorphism.
   (iv) Show that the map $s\colon F[x] \to F$ defined by $s(\sum a_i x^i) = \sum a_i$ is a ring homomorphism, and find its kernel.
3. Prove that the kernel of a ring homomorphism $\varphi\colon R \to S$ is an ideal of R.
4. Prove that if p is a polynomial in $F[x]$ such that $p(a) = 0$, then $(x-a)$ divides p. Show also that the converse is true. (*Hint: division algorithm.*)
5. Suppose R is a (commutative) ring and S,T are ideals of R. Show that $S+T = \{s+t \mid s \in S, t \in T\}$ is an ideal of R. Show that $ST = \{\sum_{i=1}^{n} s_i t_i \mid s_i \in S, t_i \in T, n \geq 0\}$ is an ideal of R.
6. In the context of the previous question: show that if S and T are principal, then so is ST.
7. The ring $F[x,y]$ is not a principal ideal domain, meaning, it has ideals which are not principal. Give an example of such an ideal. Show that there exist principal ideals S and T of $F[x,y]$ whose sum $S+T$ is not principal.
8. The ring $F[[x]]$ of formal power series in x has a unique maximal ideal. Show that every element of $F[[x]]$ which does not lie in this unique maximal ideal is a unit (that is, invertible).
9. Show that the set of ideals of a ring is a partially ordered set under inclusion, but not a totally ordered set. In particular, exhibit a ring R and two ideals I and J that are incomparable under inclusion, and exhibit a ring R with more than one maximal ideal.
10. Consider the following ideals of $\mathbb{Z}$, and draw a diagram to indicate all inclusion relations among these ideals:
$$2\mathbb{Z}, 5\mathbb{Z}, 10\mathbb{Z}, 25\mathbb{Z}, 40\mathbb{Z}.$$
    Which is/are the smallest, and which is/are the largest?
11. We need I to be an ideal in order for the multiplication in R/I to be well-defined. Show, for example, that if we take the additive subgroup $I = F$ in the ring $R = F[x]$, then cosets of I in R are of the form $p+F$ where p is a polynomial, and two polynomials are in the same coset if and only if their difference is a constant. Now find two polynomials p and q that represent the same coset but such that $p^2+F \neq q^2+F$, which shows that in this case, multiplication does depend of the choice of representative, so R/I is not a ring.
12. Consider $R = F[x]$ and let $a \in F$. Set $I = \langle x-a \rangle$. What is a set of representatives for R/I? What are the rules for addition and multiplication in this case? Do they depend on a?
13. Consider $F = \mathbb{Z}_2$, $R = F[x]$ and let $I = \langle x^2+1 \rangle$. How many elements are there in R/I? Write down a set of representatives for R/I. Write down the addition and multiplication tables for R/I. If you have seen ring isomorphisms: show this is isomorphic as a ring to $\mathbb{Z}_2 \times \mathbb{Z}_2$.
14. If $|F| = q$ and $\deg(a) = n$, how many elements are there in $F[x]/\langle a \rangle$?
15. Show that if I is an ideal of $F[x]$, where F is a field, then $F[x]/I$ is a vector space over F. If $I = \langle a \rangle$ and $\deg(a) = n$, what is the dimension of this vector space?
16. Consider Example 6.29. Consider the map $T\colon F[x]/I \to F^n$ given by $T(c_0+c_1x+\cdots+c_{n-1}x^{n-1}) = c_0c_1\cdots c_{n-1}$. (Note that we are indexing coordinates starting from 0 rather than 1.) Show this is an isomorphism of vector spaces. Multiplication by x gives an isomorphism of $F[x]/I$ to itself

so induces an isomorphism of F^n. Which one? (That is: for any vector $z \in F^n$, write down $T(x(T^{-1}(z)))$.)

7. Cyclic codes, revisited

We defined cyclic codes in Section 5.7 as subspaces of F^n which are closed under cyclic shifts.

In this chapter, we identify our codes as subspaces of the ring $F[x]/\langle x^n - 1\rangle$, as follows.

First note that $P_n = \{a_0 + \cdots + a_{n-1}x^{n-1} \mid a_i \in F\}$, the set of all polynomials of degree less than n, is a vector space of dimension n, hence isomorphic to F^n. We always choose the isomorphism

$$\pi \colon P_n \to F^n \quad \text{given by} \quad \pi(a_0 + \cdots + a_{n-1}x^{n-1}) = a_0 a_1 \cdots a_{n-1}.$$

In fact, we often suppress the notation π, fluidly saying $1 + x = 1100$ in $P_4 \cong F^4$, for example.

Next, recall that $F[x]/\langle x^n - 1\rangle$ is isomorphic to P_n as a ring, where P_n is equipped with addition and multiplication mod $x^n - 1$. As with the identification $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}_n$, we usually suppress the coset notation, writing $x^4 - 1 = 0$ in $F[x]/\langle x^4 - 1\rangle$, for example.

Thus, given $a \in F[x]$ we may think of it as representing a coset in $F[x]/\langle x^n - 1\rangle$, and thus an element of P_n, or as the corresponding vector in F^n. When helpful, we may write $a(x)$ and $\vec{a}$ to help clarify whether we are thinking of an element a as a polynomial or as a vector.

7.1 Cyclic codes as subspaces of $F[x]/\langle x^n - 1\rangle$

The key property of our identification $F[x]/\langle x^n - 1\rangle \cong F^n$ is the following.

Lemma 7.1 The first cyclic shift of a vector in F^n is given by multiplication by x in $F[x]/\langle x^n - 1\rangle$.

Proof. Consider $\vec{c} = c_0 c_1 \cdots c_{n-2} c_{n-1} \in F^n$. Then $\vec{c}^1 = c_{n-1} c_0 c_1 \cdots c_{n-2}$. Writing this as polynomials, we have

$$c(x) = c_0 + c_1x + \cdots + c_{n-2}x^{n-2} + c_{n-1}x^{n-1}$$

and thus

$$xc(x) = c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + c_{n-1}x^n = c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} \mod x^n - 1.$$

Therefore

$$x\vec{c(x)} = \vec{c}^1,$$

as required. ■

Let C be a cyclic code, viewed as a subspace of polynomials in $F[x]/\langle x^n - 1\rangle$ via the maps above. Then C being cyclic implies that the first cyclic shift of any codeword lies in C; this translates to

$$\forall c(x) \in C, \quad xc(x) \in C.$$

This implies that $x^2c(x) \in C$, and indeed $x^mc(x) \in C$ for any m; furthermore, since C is a subspace, it is closed under scalar multiplication, so $ac(x) \in C$ for any scalar a. What about multiplying by a polynomial? Let $f(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} \in F[x]/\langle x^n - 1\rangle$ be arbitrary. Then

$$f(x)c(x) = a_0c(x) + a_1xc(x) + \cdots + a_{n-1}x^{n-1}c(x)$$

is a linear combination of codewords, so is again in C, that is,

$$f(x)c(x) \in C \quad \forall f(x) \in F[x]/\langle x^n - 1\rangle.$$

This is very suggestive.

Proposition 7.2 A linear code C is cyclic if and only if it is an ideal in $F[x]/\langle x^n - 1\rangle$.

Proof. Identify $P_n \cong F[x]/\langle x^n - 1\rangle$ as at the beginning of the chapter. We showed above that if C is cyclic, then $f(x)c(x) \in C$ for all $f(x) \in P_n$. Since C is a subspace, it is closed under addition. Therefore C is an ideal of P_n.

Conversely, if I is an ideal of P_n, then it is closed under addition (being a subgroup) and scalar multiplication (since it is closed under multiplication by any element of P_n, including constants), and multiplication by $x \in P_n$. Therefore it is a subspace closed under cyclic shift, which means a cyclic code. ■

So: to classify all cyclic codes is equivalent to classifying all ideals of $F[x]/\langle x^n - 1\rangle$.

Lemma 7.3 Let R be a ring and I an ideal of R. The set of ideals of R/I is in bijection with the set of ideals of R which contain I.

Proof. Let $J \lhd R$ be such that $I \subseteq J$. Define

$$\tilde{J} = \{f + I \mid f \in J\}.$$

Note that since $I \subseteq J$, if $f \in J$ then every representative of the coset $f+I$ lies in J as well. Using this fact, it is easy to verify that $\tilde{J}$ is closed under addition and multiplication by any element of R/I, and so is an ideal of R/I.

Conversely, given an ideal $\tilde{K}$ in R/I, let K be the union of all the elements of all the cosets which are in $\tilde{K}$, that is,

$$K = \{f \in R \mid f+I \in \tilde{K}\}.$$

Again, we verify that $\tilde{K}$ being an ideal of R/I ensures that K is an ideal of R; futhermore, $I \subset K$ since in particular for each $i \in I$, $i+I = I = 0+I \in \tilde{K}$.

These two maps are inverses of one another, so the two sets are in bijection. ■

Theorem 7.4 The cyclic codes of length n are in bijection with the monic factors of x^n-1, via the correspondence:

$$C = \langle g(x)\rangle \subseteq F[x]/\langle x^n-1\rangle \quad \iff \quad h(x)g(x) = x^n-1 \quad \text{for some } h(x) \in F[x].$$

This theorem is just a summary of what we've worked out:

- cyclic codes correspond to ideals in $F[x]/\langle x^n-1\rangle$;
- the ideals in $F[x]/\langle x^n-1\rangle$ are all principal;
- such an ideal is generated by $g(x)+I$ where $g(x)$ is the generator of an ideal of $F[x]$ containing $\langle x^n-1\rangle$; and
- if $\langle g(x)\rangle \supseteq \langle x^n-1\rangle$, then $g(x)$ divides x^n-1.
- Finally, since $g(x)$ and $ug(x)$, for any $u \in F^\times$, generate the same ideal, it suffices to restrict our attention to monic polynomials.

■ **Example 7.5** Let $n=5$, and $p=2$. We have that $x^5-1 = (x-1)(x^4+x^3+x^2+x+1)$; by Example 6.13, this quartic is irreducible. Therefore the factors of x^5-1 are:

$$1, \quad x+1, \quad x^4+x^3+x^2+x+1, \quad x^5+1$$

(replacing -1 by $+1$ because we can). Let us study each of the codes in turn.

- $\langle 1\rangle$**:** The ideal generated by 1 is the whole ring, so $C_1 = F^5$.
- $\langle x+1\rangle$**:** We have $\langle x+1\rangle = \{(x+1)f \mid f \in F[x]\} \mod (x^5-1)$. As a vector space, this set is spanned by

  $$\{(x+1), \quad (x+1)x, \quad (x+1)x^2, \quad \cdots\}$$

  which is, after simplifying mod (x^5-1), the set

  $$\{1+x, \quad x+x^2, \quad x^2+x^3, \quad x^3+x^4, \quad x^4+x^5 = 1+x^4\}.$$

  Since the sum of the first 4 equals the last (over $\mathbb{Z}_2$), but the first 4 are linearly independent, we have that

  $$C_{x+1} = \text{span}\{1+x, \quad x+x^2, \quad x^2+x^3, \quad x^3+x^4\}$$

is a $(5,4)$ code. A generator matrix is

$$G_{x+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since all the codewords have even weight, and the sum of even weight codewords is even, it follows that d_{min} is even; we deduce that $d_{min} = 2$. (In fact, we see that C is exactly the set of all even weight vectors of $\mathbb{Z}_2^5$.)

- $\langle x^4+x^3+x^2+x+1\rangle$**:** Since $x(x^4+x^3+x^2+x+1) \equiv x^4+x^3+x^2+x+1 \mod (x^5-1)$, this code contains only the one nonzero vector, namely 11111. It is in fact the $(5,1)$ repetition code.
- $\langle x^5-1\rangle$: Finally, consider x^5-1. This element is 0 mod (x^5-1), so generates the zero ideal, which is a the trivial code.

We deduce that there are 4 binary cyclic codes of length 5: the trivial code, the whole space, the $(5,1)$ repetition code and the $(5,4)$ parity check code. ■

■ **Example 7.6** We saw earlier that there was a cyclic $(7,4)$ Hamming code. Let's find it here. We begin with the factorization

$$x^7-1 = (x-1)(x^3+x+1)(x^3+x^2+1),$$

and deduce there are 6 possible nontrivial cyclic codes (that is, excluding the zero code $C = \{0\}$ and the full code $C = P_n$). Our Hamming code in Example 5.26 contained the element $1101000 = 1+x+x^3 = g$, and in fact was shown to be generated by cyclic shifts of this element, so we deduce that $C = \langle 1+x+x^3\rangle$.

Note that to write down a generator matrix for C, we would proceed as above, that is, choosing a linearly independent spanning subset of $g, xg, x^2g, \cdots, x^6g$. In this case, $x^4g \in \operatorname{span}\{g, xg, x^2g, x^3g\}$ (check) and therefore these first four vectors give a basis for C. The resulting generator matrix G is

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

This is attractive, but not in standard form; but recall that since the order of the coordinates matters for cyclicity, we are not expecting our codes to be systematic. ■

■ **Example 7.7** Consider ternary cyclic codes of length 5. We have the factorization

$$x^5-1 = (x-1)(x^4+x^3+x^2+x+1) = (x+2)(x^4+x^3+x^2+x+1)$$

so there are only 4 such codes: the trivial one, corresponding to $g(x) = x^5-1 = 0$; the full codes, corresponding to $g(x) = 1$; the code with generator $g(x) = x+2$; and the code with generator $g(x) = x^4+x^3+x^2+x+1 = 11111$, which is just $\{00000, 11111, 22222\}$.

The code generated by $x+2$ has basis (exercise):

$$12000, 01200, 00120, 00012$$

and so is a $(5,4)$ code. Note that $\langle x+2\rangle$ is dual to $\langle x^4+x^3+x^2+x^1\rangle$. ■

These examples illustrate that cyclic codes are few and far between, at least if x^n-1 has few factors. This seems particularly surprising when we consider larger fields, where we might have instead expected to find more codes (since there are more subspaces altogether).

Let us consider cases where we actually have *many* factors.

■ **Example 7.8** Let $n=8$ and $p=2$. Since over $\mathbb{Z}_2$, $(x+1)^2=x^2+1$, we deduce that

$$x^8-1=(x-1)^8=(x+1)^8.$$

Therefore the distinct monic factors of x^8-1 are:

$$1,\quad x+1,\quad x^2+1,\quad (x+1)^3=x^3+x^2+x+1,\quad x^4+1,\quad (x+1)^5=x^5+x^4+x+1,$$

$$(x+1)^6=x^6+x^4+x^2+1,\quad (x+1)^7=x^7+x^6+x^5+x^4+x^3+x^2+x+1,\quad x^8+1.$$

These each correspond to cyclic $(8,k)$ codes over $\mathbb{Z}_2$. ■

This example reveals a setting where we have a rich supply of cyclic codes: when n is a power of $|F|=q$.

Proposition 7.9 Let $F=\mathbb{Z}_p$ where p is prime. Then for any $k\geq 0$,

$$x^{p^k}-1=(x-1)^{p^k}$$

is a complete factorization over F.

This proposition is true over any field F of size $q=p^\ell$ for some $\ell\geq 1$; see Chapter 8.

Proof of Proposition 7.9. First consider the case that $k=1$. Then we have

$$\begin{aligned}(x-1)^p &= \sum_{\ell=0}^{p}\binom{p}{\ell}x^{p-\ell}(-1)^\ell\\ &= x^p-px^{p-1}+\frac{p(p-1)}{2}x^{p-2}-\frac{p(p-1)(p-2)}{3!}x^{p-3}\pm\cdots\\ &\quad\cdots+\frac{p(p-1)\cdots(3)}{(p-2)!}x^2(-1)^{p-2}+px(-1)^{p-1}+(-1)^p\end{aligned}$$

Now note that every coefficient (except the first and last) has numerator divisible by p and denominator a product of factors that are strictly less than p (hence, relatively prime to p), so will be 0 mod p. If $p=2$ then $(-1)^p=1=-1$ and if p is odd then $(-1)^p=-1$. Therefore we conclude

$$(x-1)^p=x^p-1.$$

The proposition follows by a proof by induction on k (exercise). ■

In Chapter 9, we'll discover how to factor $x^n - 1$ when $\gcd(n, p) = 1$ (the setting in which we'll produce our best codes!) and thus, by putting these results together, we'll have a factorization of $x^n - 1$ over any finite field F for any $n \geq 1$.

Let's briefly explore concepts related to the factorization of $x^n - 1$, to get a sense of why there are so few of them in some cases, before returning to the exploration of these codes.

7.2 Factors of $x^n - 1$ over $\mathbb{Q}$: cyclotomic polynomials

The factorization of $x^n - 1$ over $\mathbb{Q}$ has been understood for a long time, because it relates to many great problems in number theory. Its solution allows us to deduce something about the solution over F a finite field, in some cases.

Let us begin by recalling that we can identify $\mathbb{Z}_n$ with the cyclic subgroup of $\mathbb{C}$ given by

$$\mu_n := \{e^{2\pi ij/n} \mid 0 \leq j < n\},$$

which are just points on the unit circle (where $i = \sqrt{-1}$). The additive group $\mathbb{Z}_n$ is isomorphic to the multiplicative group μ_n, via the map that sends x to $e^{2\pi ix/n}$. Let M_n be the subset of μ_n corresponding to $\mathbb{Z}_n^\times$; thus

$$M_n = \{e^{2\pi ij/n} \mid \gcd(j, n) = 1\}.$$

The polynomial

$$\Phi_n(x) = \prod_{\zeta \in M_n} (x - \zeta)$$

is called the nth *cyclotomic polynomial*. We define $\Phi_1(x) = x - 1$.

What is utterly astonishing, but kind of the whole point, is that even though the definition is complex (in both senses of the word), the resulting polynomial Φ_n is quite simple.

■ **Example 7.10** By direct calculation, you can confirm that

$$\Phi_2(x) = x + 1, \qquad \Phi_3(x) = x^2 + x + 1, \qquad \text{and} \qquad \Phi_4(x) = x^2 + 1.$$

■

Lemma 7.11 If p is a prime then $\Phi_p(x) = x^{p-1} + x^{p-2} + \cdots + 1$.

Proof. If p is prime then $M_p = \mu_p \setminus \{1\}$. The roots of $x^p - 1$ are exactly the p elements in μ_p, so dividing by that one extra root we have

$$\Phi_p(x) = \frac{x^p - 1}{x - 1} = x^{p-1} + x^{p-2} + \cdots + 1.$$

■

Notice from these examples that although these polynomials have non-real roots, all their coefficients are in $\mathbb{Z}$. This is not an accident!

Theorem 7.12 Let $n \geq 1$. Then $\Phi_n(x) \in \mathbb{Z}[x]$, and is an irreducible polynomial over $\mathbb{Q}[x]$. Furthermore, we have

$$x^n - 1 = \prod_{d|n} \Phi_d(x), \tag{7.1}$$

where the product runs over all positive divisors of n.

The proof of this result requires more information about extension fields than we currently have at our disposal so we will not include it here; see, for example, [Nic12, §10.4].

Nevertheless, some thought reveals that (7.1) clearly holds. What is amazing is that the polynomials have integer coefficients, which implies that we may map these polynomials into $\mathbb{Z}_p[x]$ for any prime p, and obtain a factorization of $x^n - 1$ in the ring $\mathbb{Z}_p[x]$.

That said, there is no reason for the cyclotomic polynomials to remain irreducible when we consider them in $\mathbb{Z}_p[x]$.

■ **Example 7.13** In $\mathbb{Z}_2[x]$ we have $\Phi_4(x) = (x-1)^2$, so it is not irreducible, even though, according to the theorem, $\Phi_4(x)$ is irreducible when considered as a polynomial in $\mathbb{Z}[x]$. ■

This theorem therefore gives us a first rough factorization of $x^n - 1$, over any field F. [1]

Many cyclotomic polynomials have been calculated, and they satisfy many relations which make them relatively easy to work out; see for example the article on Wolfram Mathworld.

7.3 Another constraint on cyclic codes over $\mathbb{Z}_p$

In the previous section, we had some indication that $x^n - 1$ factors in a very special way, at least over $\mathbb{Q}$. Now let us see what we can say about this factorization over $\mathbb{Z}_p$.

If C is a cyclic code, then by definition it is closed under cyclic shifts, that is, if $c_0 c_1 \cdots c_{n-1} \in C$ then so is $c_{n-1} c_0 \cdots c_{n-2}$; this was a consequence of multiplying the polynomial form of c by x in $F[x]/\langle x^n - 1\rangle$.

But we also have, over $\mathbb{Z}_p$, a fascinating identity. There is a subtle point in its statement and proof. The variable x is an indeterminate, so we **do not have** that $x^p = x$. One way to see this is to consider that although the polynomial x^p gives the same answer as x whenever we evaluate on elements of $\mathbb{Z}_p$, we could evaluate it at any $a \in F$ where F is a field containing $\mathbb{Z}_p$, where $a^p \neq a$. Therefore as functions on this extension field, x^p is not the same as x. We reconcile this weirdness by stating that $F[x]$ is the ring of *formal polynomials* over F, to distinguish it from the set of polynomial functions on F.

Lemma 7.14 Let $a(x) \in \mathbb{Z}_p[x]$. Then $(a(x))^p = a(x^p)$ as polynomials.

[1] More generally, one can give a definition of $\Phi_n(x)$ that makes sense over any field, such as is done in [Nic12, §10.4], and more subtleties of the relation between the characteristic of F and n come into play.

Proof. Begin by noticing that (exercise) for any $x, y \in \mathbb{Z}_p$, we have

$$(x+y)^p = x^p + y^p.$$

Furthermore, by Fermat's little theorem, we have that $a^p = a$ for any $a \in \mathbb{Z}_p$. So proceeding inductively, we have that

$$\begin{aligned}
(a(x))^p &= (a_0 + a_1 x + \cdots + a_{n-1}x^{n-1})^p \\
&= (a_0 + \left(\sum_{i=1}^{n-1} a_i x^i\right))^p \\
&= a_0^p + \left(\sum_{i=1}^{n-1} a_i x^i\right)^p \quad \text{binomial case} \\
&= a_0^p + (a_1 x)^p + \cdots + (a_{n-1}x^{n-1})^p \quad \text{by induction} \\
&= a_0^p + a_1^p x^p + \cdots + a_{n-1}^p x^{(n-1)p} \\
&= a_0 + a_1 x^p + \cdots + a_{n-1}x^{(n-1)p} \quad \text{each } a_i \in \mathbb{Z}_p \\
&= a(x^p).
\end{aligned}$$

■

Proposition 7.15 Suppose C is a cyclic code of length n over $\mathbb{Z}_p$, where $\gcd(n,p) = 1$. Then C is closed under the permutation $\sigma \in S_n$ which sends i to $ip \mod n$.

Proof. Since $\gcd(p,n) = 1$, p is invertible mod n so σ is a bijective map on the set $\{0, 1, \cdots, n-1\}$, hence indeed in S_n.

Since C is a cyclic code, there exists some $g(x)$ such that $C = \langle g(x) \rangle$. Suppose $a(x) \in C$. Then there is some $q(x)$ such that $a(x) = q(x)g(x)$. Then by the lemma we have

$$a(x^p) = a(x)^p = (q(x)g(x))^p = (q(x)^p g(x)^{p-1})g(x)$$

which is again a multiple of g, so in C. If $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, then $a(x^p) = b_0 + b_1 x + \cdots + b_{n-1}x^{n-1}$ where $b_0 = a_0$ and for each $i > 0$, $b_{pi \mod n} = a_i$ (since a_i is the coefficient of x^{pi} in $a(x^p)$). This is the map $\sigma^{-1}(a_0 a_1 \cdots a_{n-1}) = a_{\sigma^{-1}0} a_{\sigma^{-1}1} \cdots a_{\sigma^{-1}(n-1)} = b_0 b_1 \cdots b_{n-1}$. So C is closed under σ^{-1}. Since S_n is finite, σ has finite order, so there is some k such that $(\sigma^{-1})^k = \sigma$, and so the lemma follows. ■

■ **Example 7.16** Suppose $n = 7$ and $p = 2$. The theorem says that if $c_0c_1c_2c_3c_4c_5c_6 \in C$ then so is $c_0c_2c_4c_6c_1c_3c_5$; the proof showed us that the inverse permutation $c_0c_4c_1c_5c_2c_6c_3$ is also in C. For example, in the cyclic Hamming code discussed above and in Example 5.26, we have $1101000 \in C$; its σ-twist is 1000110; the inverse twist is 1010001; both of these are just cyclic shifts of 1101000 so are obviously in C. ■

■ **Example 7.17** Consider the ternary cyclic codes of length 5 discussed in Example 7.7. Since $3 \times 1 = 3$, $3 \times 2 = 1$, $3 \times 3 = 4$ and $3 \times 4 = 2$, the theorem says that any such code is closed under

$c_0c_1c_2c_3c_4 \mapsto c_0c_3c_1c_4c_2$. This is obvious for three of the four codes in that example; for the last (the one generated by 12000) we compute

$$12000 \mapsto 10200 = 12000 + 01200 \in C$$

for example. ■

The theorem tells us that the requirement of a linear code being cyclic implies more constraints on the coordinates than we perhaps intended, because of these interactions with the finite field. This is reflected in the complexity of the solution to the question of how many cyclic codes there are over any given field.

7.4 Exercises

1. Complete the details of the proof of Lemma 7.3.
2. Suppose $I = \langle 10 \rangle$ and $J = \langle 6 \rangle$. What is $\tilde{J}$ (notation as in the proof of Lemma 7.3)?
3. Write out the details of the proof of Theorem 7.4.
4. Factor $x^3 - 1$ into irreducibles over $\mathbb{Z}_2$ and use this to give a list of all ideals in $\mathbb{Z}_2[x]/\langle x^3 - 1 \rangle$. Use this to write down generator matrices for all cyclic codes in $\mathbb{Z}_2^3$.
5. Complete the proof of Proposition 7.9, by doing the argument by induction.
6. Prove that over a field F of characteristic p, where p is of course prime, we have

$$(x+y)^p = x^p + y^p$$

for all $x, y \in F$. (For those who have not seen "characteristic": prove this for the case that $F = \mathbb{Z}_p$.)
7. Prove that if $F = \mathbb{Z}_p$, then $a^p = a$ for all $a \in F$. Hint: this is close to Fermat's little theorem. (The pth-power map becomes interesting if F is a field of characteristic p which is not equal to $\mathbb{Z}_p$; it is sometimes called a *Frobenius map*.)
8. Find the dimension of and generator matrices for each of the cyclic codes in Example 7.8.
9. Describe the remaining binary cyclic codes of length 7, given the factorization of $x^7 - 1$ in Example 7.6.
10. Prove directly that the generator matrix of Example 7.6 is that of a Hamming code.
11. Find generators for all ternary cyclic codes of length 4.
12. Show that the cyclic code generated by 11000 over $\mathbb{Z}_3$ is the full code $\mathbb{Z}_3^5$. (Note that 11000 means $1 + x$ and that $x + 1 \neq x - 1 = 21000$ over $\mathbb{Z}_3$. It's important to remember that the first factor of $x^n - 1$ is $x - 1$; it's only over binary fields that we can pretend it is $x + 1$.)
13. Find all factors of $x^9 - 1$ over $\mathbb{Z}_3$. Which factors will generate bigger cyclic codes, and which smaller? Find a basis for the code generated by $(x-1)^7$.
14. Suppose $g(x) \in F[x]$ is an irreducible polynomial that does not divide $x^n - 1$. What is $\langle g(x) \rangle$?
15. Find the full group of permutation automorphisms of a cyclic binary code of length 5. Generalize.
16. Work through the argument of the proof of Proposition 7.15 in the case that $n = 6$ and $p = 2$, and identify what makes this case different. What does the closure under $a(x) \mapsto a(x^p)$ give you in this case?

7.5 A first generator matrix for a cyclic code

So we've classified all cyclic codes: the cyclic codes of length n over F are in one-to-one correspondence with monic factors of x^n-1 in $F[x]$. But how big is such a code?

Theorem 7.18 Let g be a monic factor of x^n-1 and set $C=\langle g\rangle$. If $\deg(g)=n-k$ then C is an (n,k) cyclic code.

Proof. We will find a basis for C, which in turn allows us to determine its dimension, and, as a corollary, a generator matrix for C.

First, since g divides x^n-1 we may set

$$h=\frac{x^n-1}{g}\in F[x];$$

since $\deg(g)+\deg(h)=n$, we have that $\deg(h)=k$.

An element of C is of the form ag for some polynomial $a=\sum_i a_i x^i$. If $\deg(a)<k$, then this implies $ag\in \operatorname{span}\{g,xg,\cdots,x^{k-1}g\}$. Otherwise, we first use long division to write $a=qh+r$ with $\deg(r)<\deg(h)$; then since $hg=x^n-1$ we have

$$ag=(qh+r)g=q(hg)+rg\equiv rg \mod (x^n-1)$$

which shows that $ag=rg\in\operatorname{span}\{g,xg,\cdots,x^{k-1}g\}$. Thus $C=\operatorname{span}\{g,xg,\cdots,x^{k-1}g\}$.

To see that these vectors are linearly independent, it suffices to note that if $rg\equiv 0 \mod (x^n-1)$ then necessarily $\deg(r)+\deg(g)\geq n$, which implies $\deg(r)\geq k$. So no nontrivial dependence relation can exist.

We conclude that $\{g,xg,\cdots,x^{k-1}g\}$ is a basis for C and that $\dim(C)=k$. ■

R Another way of thinking about the proof is the following, which is related to how we'll decode, later. Begin by noting that since g generates C and $x^n g=g$, the set

$$S=\{g,xg,x^2g,\cdots,x^{n-1}g\}$$

spans C. Say $h=h_0+h_1x+\cdots h_kx^k$. The equation $hg=x^n-1$ becomes the equivalence

$$h_0g+h_1xg+\cdots+h_kx^kg\equiv 0 \mod (x^n-1)$$

which is a nontrivial dependence relation on some of the elements of S. In fact, since $h_k\neq 0$ (by definition of degree), we deduce that x^kg is expressible as a linear combination of $\{g,xg,\cdots,x^{k-1}g\}$. Multiplying by x and repeating the process, we see that indeed $\{g,xg,\cdots x^{k-1}g\}$ is enough.

Suppose $g(x)=g_0+g_1x+\cdots+g_{n-k}x^{n-k}$. Then one generator matrix for C is given by taking the first column to be $g_0g_1\cdots g_{n-k}0\cdots 0$ and the next $k-1$ columns to be its cyclic shifts (ending with a column

$0\cdots 0 g_0 g_1 \cdots g_{n-k}$):

$$G = \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 & 0 \\ g_1 & g_0 & 0 & \cdots & 0 & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & g_{n-k} & g_{n-k-1} \\ \vdots & \vdots & \vdots & \cdots & 0 & g_{n-k} \end{bmatrix} \tag{7.2}$$

With such a matrix, the message $a \in F^k$ is encoded as Ga; in polynomials, $a(x)$ is a polynomial of degree $\leq k-1$, and (identifying vectors with polynomials in our usual way) we have

$$Ga = a(x)g(x) = g(x)a(x). \tag{7.3}$$

That is, we can compute the encoding with respect to G without using matrix multiplication; use polynomial multiplication instead.

■ **Example 7.19** Consider $F = \mathbb{Z}_2$ and $n = 8$, as in Example 7.8. Ignoring the zero code, we have exactly one cyclic code of dimension k for each k between 1 and 8, since there are factors of $x^8 - 1$ of each degree. Let's work out one of them.

Consider $g = (x+1)^5 = x^5 + x^4 + x + 1 = g_5x^5 + g_4x^4 + g_3x^3 + g_2x^2 + g_1x + g_0$. Since $\deg(g) = 5$ and $n = 8$, the code generated by g has dimension $8 - 5 = 3$ and a basis is $\{g, xg, x^2g\}$. Notice that since $\deg(x^2g) = 7 < 8$, we will not need to reduce modulo $x^8 - 1$ to compute these basis vectors! They are

$$\{g(x) = 1 + x + x^4 + x^5, \quad xg(x) = x + x^2 + x^5 + x^6, \quad x^2g(x) = x^2 + x^3 + x^6 + x^7\}.$$

Written as vectors this basis is

$$\{\vec{g} = 11001100, \quad \vec{xg} = 01100110, \quad \vec{x^2g} = 00110011\}$$

so a generator matrix is

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

which is exactly as predicted by (7.2) just above. ■

7.6 Parity check matrices for cyclic codes

Now we go on to find a parity check matrix. We are looking for a matrix H of rank $n-k$ such that $HG = 0$. In the proof of Theorem 7.18, we saw that if h is the polynomial of degree k such that $hg = x^n - 1$, then $hg \equiv 0 \mod \langle x^n - 1 \rangle$. Thus we naturally expect that H has something to do with h. The surprise is that we have to write h backwards.

Proposition 7.20 Let g be a monic factor of $x^n - 1$ and set $C = \langle g \rangle$. If $gh = x^n - 1$ and $h = \sum_{i=0}^{k} h_i x^i$, then a parity check matrix for C is given by the matrix with rows

$$h_k h_{k-1} \cdots h_1 h_0 0 \cdots 0$$

and its $n-k$ cyclic shifts, ending with $0 \cdots 0 h_k \cdots h_1 h_0$, that is,

$$H = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_1 & h_0 & 0 & \cdots & 0 \\ 0 & h_k & \cdots & h_2 & h_1 & h_0 & \cdots & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & \cdots & \cdots & h_2 & h_1 & h_0 \end{bmatrix}.$$

Proof. It is easy to see that this matrix is of size $(n-k) \times n$ and has full rank (since both h_k and h_0 are necessarily nonzero (exercise)), so it suffices to verify that $HG = 0$.

To better understand what the product HG represents, let's write

$$g(x) = g_0 + g_1 x + \cdots + g_{n-1} x^{n-1}$$

where $g_i = 0$ for $i > n-k$, and write $h(x) = \sum_{i=1}^{n-1} h_i x^i$. Then the columns of G are the cyclic shifts of

$$g_0 g_1 g_2 \cdots g_{n-2} g_{n-1}$$

whereas the rows of H are various cyclic shifts of the vector

$$h_{n-1} h_{n-2} \cdots h_1 h_0.$$

Now, what does the dot product of these vectors have to do with the product $h(x)g(x)$?

We multiply:

$$\begin{aligned} g(x)h(x) &= (\sum_{i=0}^{n-1} g_i x^i)(\sum_{j=0}^{n-1} h_j x^j) \\ &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} g_i h_j x^{i+j} \\ &= \sum_{r=0}^{n-1} \left(\sum_{i=0}^{n-1} g_i h_{r-i} \right) x^r \end{aligned}$$

where all subscripts are interpreted as mod n. Reducing $g(x)h(x)$ modulo $x^n - 1$ only affects the coefficients of x^0 and x^n, since all other powers of x are less than n; we find that the constant term of $g(x)h(x) \bmod x^n - 1$ is (by removing all coefficients that are automatically zero) $g_0 h_0 + g_{n-k} h_k$.

Since a polynomial is zero iff each of its coefficients is, we obtain that $g_0 h_0 + g_{n-k} h_k = 0$ and the $n-1$ equations

$$\sum_{i=0}^{n-1} g_i h_{r-i \mod n} = 0, \quad \forall 0 < r < n.$$

This is precisely the statement that every row of H times every column of G is zero. We conclude that $HG = 0$. ■

! Suppose $g(x)h(x) = x^n - 1$. We could set $C = \langle g \rangle$ and $D = \langle h \rangle$. What the proof of the theorem warns us is: even though $\dim(C) + \dim(D) = n$, and $gh \equiv 0$, $D = \langle h \rangle$ need *not* be the dual code of $C = \langle g \rangle$.

Define the *reciprocal polynomial* of h as the polynomial

$$h'(x) = x^k h(x^{-1}) = x^k(h_0 + h_1 x^{-1} + \cdots + h_{k-1}x^{-(k-1)} + h_k x^{-k}) = h_k + h_{k-1}x + \cdots + h_0 x^k.$$

Then the proof shows us that $C^\perp = \langle h' \rangle$. Thus we have $C^\perp = \langle h \rangle$ if and only if $\langle h \rangle = \langle h' \rangle$ (such as when $h(x)$ is a palindrome).

Thus we have a first construction of a generator matrix G and a parity check matrix H for our cyclic code. The nice feature is that we can easily write down both from the factorization of $x^n - 1$. There are other perks as well, as we'll see next.

7.7 Alternate generator matrix

Although we've found a generator matrix for our cyclic code, we notice that it is not in standard form. For cyclic codes, we cannot simply apply the arguments that show each equivalence class of codes admits a systematic code, since permuting the coefficients could destroy the cyclic nature of the code.

On the other hand, we might suspect that a more efficient choice is nevertheless available, since in the proof that $HG = 0$ (which consists of $(n-k)k$ dot products equalling zero) we saw that we could compute this from the product of two polynomials (with only n homogeneous equations). That is, one single coefficient of the product $g(x)h(x)$ determined several of the entries of the matrix HG.

With these thoughts in mind, let us find another generator matrix for a cyclic code.

Our idea is the following. Suppose $C = \langle g(x) \rangle$, with $\deg(g) = n-k$, as always. Let $n-k \le i < n$. Then we can use the division algorithm to write

$$x^i = q_i(x)g + r_i(x)$$

and $\deg(r_i(x)) < n-k$. From this equation, we deduce that

$$x^i - r_i(x) \in C$$

for each of the k values of i. These polynomials are clearly linearly independent, since each one contains a unique monomial of degree larger than $n-k-1$; there are k polynomials, and they all lie in C, so this is a basis for C. The corresponding generator matrix is

$$G' = \begin{bmatrix} -R \\ I \end{bmatrix} \tag{7.4}$$

where R is the matrix whose columns are the coefficients of $r_{n-k}(x)$, $r_{n-k+1}(x)$, $\cdots$, $r_{n-1}(x)$. This is still not standard, but it has all the same advantages as a standard matrix (and more!).

Having a different generator matrix implies we have a choice of how to calculate the encoding of $a \in F^k$.

Lemma 7.21 Let $a \in F^k$, and identify this with the polynomial $a(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}$. Let $r(x)$ be the remainder of division of $a(x)x^{n-k}$ by $g(x)$. Then (identifying vectors and polynomials in our usual way) we have

$$G'a = a(x)x^{n-k} - r(x),$$

that is, we can compute the encoding with respect to G' without using matrix multiplication.

Proof. Let us compute $G'a$. We compute this matrix product as $\sum_{i=0}^{k-1} a_i g^i$, where g^i is the ith column of G'. In polynomial form, this sum is simply

$$G'a = \sum_{i=0}^{k-1} a_i (x^{n-k+i} - r_{n-k+i}(x)) = x^{n-k} \sum a_i x^i - \sum_{i=0}^{k-1} a_i r_{n-k+i}(x) = x^{n-k} a(x) - r'(x).$$

Note that $r'(x)$ is a polynomial of degree less than $n-k$ (since each of the $r_i(x)$ are). Now on the other hand, $G'a \in C = \langle g(x) \rangle$, so there exists some $q(x)$ such that $G'a = q(x)g(x) \mod \langle x^n - 1 \rangle$. Since $g(x)$ divides $x^n - 1$, this means $G'a = q'(x)g(x)$ for some $q'(x)$. Thus we have an equation

$$a(x)x^{n-k} = q'(x)g(x) + r'(x)$$

with $\deg(r'(x)) < n-k$. But the division theorem tells us that such an expression is *unique*. In other words, $r'(x)$ is precisely the remainder of $a(x)x^{n-k}$ upon division by $g(x)$. ■

Thus, the effect of using G' in place of G is to encode words using division by $g(x)$ (Lemma 7.21) rather than multiplication by $g(x)$ (as in (7.3)). There exist more efficient algorithms for the division operation than for multiplication, so G' is preferred.

It is also easier to decode $a(x)$ from its codeword: just take the last k bits.

■ **Example 7.22** Consider $F = \mathbb{Z}_2$ and $n = 7$. Then we have

$$x^7 - 1 = (x-1)(x^3 + x + 1)(x^3 + x^2 + 1).$$

Let us take the cyclic code C generated by

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

which has degree $n - k = 4$; thus $\dim(C) = k = 3$. To compute G', we first need to divide each of $x^{n-k}, \cdots, x^{n-1}$ (that is, x^4, x^5, x^6) by $g(x)$. We can do so recursively with little effort:

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

so multiplying by x and simplifying again yields

$$x^5 = xg(x) + x^4 + x^3 + x = xg(x) + (g(x) + x^3 + x^2 + 1) + x^3 + x = (1+x)g(x) + x^2 + x + 1$$

and even more simply

$$x^6 = (x^2 + x)g(x) + x^3 + x^2 + x.$$

At each step, we have used the uniqueness of the remainder to be assured that since our alleged remainder has degree less than $\deg(g)$, it is the actual remainder upon division by $g(x)$.

Thus we have $r_4(x) = x^3 + x^2 + 1$, $r_5(x) = x^2 + x + 1$ and $r_6(x) = x^3 + x^2 + x$, which means

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

Now let us encode the message $a = 101$ using Lemma 7.21.

To do so, begin by writing $a(x) = 1 + x^2$. Then $x^{n-k}a(x) = x^4 a(x) = x^4 + x^6$. We do long division (over $\mathbb{Z}$, for ease of reading):

$$\begin{array}{r|rrrrrrr}
\multicolumn{1}{r}{} & & & & & x^2 & -x & +1 \\ \cline{2-8}
x^4+x^3+x^2+1 & x^6 & & +x^4 & & & & \\
 & -x^6 & -x^5 & -x^4 & & -x^2 & & \\ \cline{2-6}
 & & -x^5 & & & -x^2 & & \\
 & & x^5 & +x^4 & +x^3 & & +x & \\ \cline{3-7}
 & & & x^4 & +x^3 & -x^2 & +x & \\
 & & & -x^4 & -x^3 & -x^2 & & -1 \\ \cline{4-8}
 & & & & & -2x^2 & +x & -1
\end{array}$$

which means (over F) that

$$x^6 + x^4 = (x^2 + x + 1)g(x) + x + 1.$$

Thus $r(x) = x + 1$, and therefore our codeword is

$$G'a = a(x)x^{n-k} - r(x) = x^6 + x^4 + x + 1 = 1 + x + 0x^2 + 0x^3 + x^4 + 0x^5 + x^6 = 1100101.$$

We check that in fact, yes, this is the same vector we would get if we calculated the matrix product $G'a$.
■

Lemma 7.23 Let C and R be as in Lemma 7.21. Then the matrix

$$G'' = \begin{bmatrix} I \\ -R \end{bmatrix}$$

is a standard generator matrix for C.

Proof. Since G' is a generator matrix for C, its columns form a basis B for C. Since C is cyclic, the cyclic shift is an automorphism of C. Applying the cyclic shift k times to each vector in B yields therefore another basis B^k of C; these are precisely the columns of G''. It is clear than G'' is in standard form. ■

We deduce a surprising consequence.

Corollary 7.24 All cyclic codes are systematic.

It turns out that G'' gives the most efficient encoding algorithm of all.

7.8 Encoding cyclic codes with shift registers

A good reference for the material in this section is [Ple98, Ch 5]. In this section, we work only over the binary field; this decoding algorithm is specific to the case of $\mathbb{Z}_2$.

Let C be a cyclic (n,k) code over the binary field, with generator polynomial $g(x)$. Define $h(x)$ and H as in Proposition 7.20 and G'' is a standard generator matrix as in Lemma 7.23. Note that since H is a parity check matrix for C, we have $HG'' = 0$ (and in fact $HG = 0$ where G is as in (7.2), and $HG' = 0$ where G' is as in (7.4)).

Goal:

> We will use the structure of H to implement an *encoding* algorithm to compute the codeword $G''a \in F^n$ from $a \in F^k$ more efficiently than matrix multiplication.

To understand the algorithm, let $a \in F^k$, and recall that $h(x)$ is a polynomial of degree k (so is one bit longer than a). We wish to find $G''a$, which is the systematic encoding of a. This a vector $c \in C$ such that

$$c = a_0 a_1 \cdots a_{k-1} c_k c_{k+1} \cdots c_{n-1}$$

for some $c_k, \ldots, c_{n-1}$ to be determined.

Since H is a parity check matrix for C, we must have $Hc = 0$. Consider the first coordinate of $Hc = 0$. This yields the equation

$$h_k a_0 + h_{k-1} a_1 + \cdots + h_1 a_{k-1} + h_0 c_k = 0$$

so (since $h_0 = 1$, since it's a factor of $x^n - 1$) we can solve for c_k. If we next consider the second entry of $Hc = 0$, we get the equation

$$h_k a_1 + h_{k-1} a_2 + \cdots + h_2 a_{k-1} + h_1 c_k + h_0 c_{k+1} = 0$$

which allows us to solve for c_{k+1}. We can then proceed inductively through the rows of H, computing one more coefficient of c each time.

How this is implemented on a computer is through a *shift register*. This is a logical unit consisting of (in our case) k flip-flops, connected in such a way that the following is true:

- We begin by loading a into the shift register (one bit per flip-flop).
- The shift register can move all data one unit *left* per clock cycle, dumping the leftmost bit out (into a serial stream, for example).
- We can grab the data that is moving left and put it through a binary adder and direct the sum to the new, open, empty space on the right.

In our case, what we implement on the last step is the sum

$$c_k = \sum_{i \mid h_{k-i+1} \neq 0} a_i.$$

(We can do this because h is fixed throughout.) The result of one clock cycle is therefore

$$a_1 \cdots a_k \mapsto a_1 \cdots a_k c_k$$

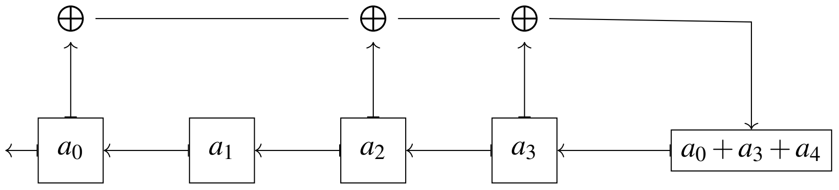
and an output of the bit a_0. Repeating this operation is equivalent to solving for c_{k+1} using the second row of H. Thus after n clock cycles, we have encoded and output the codeword for a.

We did not use G'' at all in this computation, but it can be shown (exercise) that c must be $G''a$.

■ **Example 7.25** Let $n = 7$, $p = 2$ and $g(x) = 1+x+x^3$. Then $h(x) = (x+1)(1+x^2+x^3) = 1+x+x^2+x^4$. So the parity check matrix H is

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

We set up our shift register to add the first, third and fourth elements together. (One draws this as 4 squares, representing your flip-flops, and directed lines to indicate the flow; addition is achieved through XOR gates, whose symbol is $\oplus$.)



Then if we input $a = 1010$ we get the following outputs (where we leave our leftmost bits visible, for clarity):

- 1010
- 1 0100
- 10 1000
- 101 0001
- 1010 0011
- 10100 0110
- 101000 1101
- 1010001 1010 (stop)

Notice that our register contains the original data again; this would just keep repeating if we continued.

It is easy to verify that $c = 1010001 \in C$, by computing $Hc = 0$. ■

Notice that the shift register can be used for error detection as well, since entering any portion of the codeword should generate the rest of the codeword.

It also therefore allows a rudimentary (but fast) single-error correction: if your received word w has only one error, then there will be a contiguous subset of k bits of w which are correct; feeding them through the shift register would then produce a codeword (after cyclic shift) which coincides with w in all but one bit. Applying the shift register to a subset of bits that contained an error, on the other hand, would compound the errors and produce a codeword that differed from w in several bits. Thus applying the shift register to random selections of k continguous bits of your received word would eventually reveal the codeword (as the one that differed from w in only one bit).

However, if you cannot be certain that there exists a contiguous set of k correct bits, this method is not equivalent to finding the nearest codeword.

Where to next?

We've established that cyclic codes have uniquely wonderful encoding properties, which make them useful in practice — but we haven't yet answered the most important question!

What is d_{min}? We might have hoped that working out a parity check matrix would allow us to determine d_{min} in some clever way, but we don't really see anything obvious about the smallest set of linearly dependent columns. It will turn out (preview of things to come) that the answer has to do with how many consecutive roots our polynomial has — not something we're prepared to tackle, yet.

In the next chapter, we move on to the construction of finite fields, and an analysis of cyclotomic polynomials over finite fields which is different, in practice, from the complex case we discussed earlier.

7.9 Exercises

1. Show that if $hg = x^n - 1$ then the constant terms of h and g are in $F^\times$.
2. Suppose $g = g_0 + g_1 x + \cdots + g_r x^r$ is a polynomial of degree r that divides $x^n - 1$. Prove that $h = g_r + g_{r-1}x + \cdots + g_1 x^{r-1} + g_0 x^r$ also divides $x^n - 1$. (*Hint: one way is to prove and use the formula $h(x) = x^r g(x^{-1})$. Another is to remember that the dual of a cyclic code is cyclic, and use the results of this chapter.*)
3. Use the methods of Section 7.7 to produce a standard generator matrix and corresponding parity check matrix for the binary $(8,3)$ cyclic code.
4. Give a direct proof of Lemma 7.21. That is, show that $a(x)x^{n-k}$ is a linear combination of $\{x^i \mid n-k \le i < n\}$ and that therefore the remainder $r(x)$ of the theorem is precisely the same linear combination of the remainders $\{r_i(x) \mid n-k \le i < n\}$. Infer that the decoding algorithm is correct.
5. Describe an algorithm to divide a binary polynomial of degree n by a binary polynomial of degree k. Estimate the number of operations required. Compare this with the cost of multiplying a polynomial of degree k with one of degree $n-k$. Compare with the shift register.
6. Show that the output of the shift register encoding algorithm is $G''a$. *Hint: prove that there is only one codeword in C which has a as a prefix.*
7. Compute the encoding of 1110 using the shift register for the $(7,4)$ Hamming code.
8. Define the shift register for your favourite binary cyclic code of length 9.

8. Finite fields, beyond $\mathbb{Z}_p$

We have previously seen that for every prime p, $\mathbb{Z}_p$ is a field. In this chapter, we use what we learned about quotients of polynomial rings to construct all finite fields. We'll then relate this back to our unanswered questions about cyclic codes.

A reference for the material about field extensions in this section is [Nic12, Chap 6.2, 6.3].

8.1 Looking for other finite fields

Definition 8.1 The *characteristic* of a field F is the least positive integer n such that $1+1+\cdots+1$ (n times) equals 0. If no such n exists, then we define the characteristic to be $n=0$. In any case, we write $n=\operatorname{char}(F)$.

■ **Example 8.2** We have $\operatorname{char}(\mathbb{Z}_5)=5$ since $1+1+1+1+1\equiv 0 \mod 5$, and no subsum is zero.

We have $\operatorname{char}(\mathbb{R})=0$ since $1+1+\cdots$ is never 0. ■

It is clear that *if* the characteristic of a field is zero, then the field is infinite. (But the converse is not true! There are fields of positive characteristic which are infinite, like the field of infinite Laurent polynomials in one variable with coefficients in $\mathbb{Z}_p$.) Put another way: any finite field has positive characteristic.

Lemma 8.3 If F is a field, then $\operatorname{char}(F)$ is either 0 or a prime.

Proof. Suppose $\operatorname{char}(F)=n>0$. If n has a nontrivial factorization as $n=ab$, with $1<a,b<\operatorname{char}(F)$, then first write

$$a=1+1+\cdots+1\ (a\text{ times})\quad\text{and}\quad b=1+1+\cdots+1\ (b\text{ times}).$$

Then by distributivity, ab is the sum of ab 1s, which is 0 since $ab=n=\operatorname{char}(F)$. But F is a field, so

$ab = 0$ implies one of a or b is zero. But both a and b are sums of fewer than n 1s, so are not zero, a contradiction. We conclude that n is prime. ■

This is really quite handy. For instance, suppose F is a finite field of characteristic p. Then we obtain a well-defined map

$$\varphi : \mathbb{Z} \to F \quad \varphi(n) = 1 + 1 + \cdots + 1 (n \text{ times})$$

whose kernel is $p\mathbb{Z}$. Thus $\mathbb{Z}/p\mathbb{Z}$, which is what we call $\mathbb{Z}_p$, is isomorphic to the image of φ in F. In other words, with this identification, we have that $\mathbb{Z}_p \subseteq F$.

Theorem 8.4 If F is a field of characteristic p, then it is a vector space over the field $\mathbb{Z}_p$.

We leave the proof of Theorem 8.4 as an exercise. Note that the operations of the vector space are just a subset of the operations of the field.

So if F is a finite field of characteristic p, then F is a (finite-dimensional) vector space over $\mathbb{Z}_p$. Let $\{v_1, \cdots, v_n\}$ be a basis for this vector space; then every element of F is uniquely expressible in the form

$$a_1 v_1 + \cdots + a_n v_n$$

for some $a_1, \cdots, a_n \in \mathbb{Z}_p$. This proves an astounding fact.

Corollary 8.5 If F is a finite field, then $|F| = p^n$ for some prime p and some $n \in \mathbb{N}$.

In fact, there is a deeper theorem which asserts: for each prime p and each $n \in \mathbb{N}$, there exists up to isomorphism a unique field of order p^n. We sometimes call it the *Galois field* of order p^n, denoted $GF(p^n)$ or $\mathbb{R}_{p^n}$.

8.2 Using quotient rings to construct fields

We have previously seen that for any ideal I, R/I is again a ring. When is this a field?

Well a (commutative, unital) ring is a field exactly when every nonzero element is invertible. By Lemma 6.22, this implies the ring has no proper ideals except $\{0\}$. By Lemma 7.3, the ideals of R/I are exactly the ideals of R containing I.

Therefore: R/I is a field when the only ideals of R containing I are R and I. We give such ideals a name.

Definition 8.6 A *maximal ideal* I of a ring R is a proper ideal such that if J is any other ideal of R satisfying $I \subseteq J \subseteq R$ then either $J = I$ or $J = R$.

OK: so to make R/I into a field, we need to choose I to be a maximal ideal. So what are the maximal ideals of a ring like $F[x]$, in which all ideals are principal? Well, suppose $\langle a \rangle$ is a maximal ideal of R.

Theorem 8.7 Let R be either $\mathbb{Z}$ or $F[x]$ for some field F. Let $I = \langle a \rangle$ be a maximal ideal of R. Then a is irreducible and $R/\langle a \rangle$ is a field.

Proof. All ideals of R are principal, by Theorem 6.21. So suppose I is a maximal ideal and let $a \in R$ be such that $I = \langle a \rangle$. Then $a \notin R^\times$ (or else $\langle a \rangle = R$), and by Lemma 6.23,

$$\langle a \rangle \subseteq \langle b \rangle \qquad \Longleftrightarrow \qquad b \text{ divides } a$$

so a maximal ideal corresponds to an element a which has no nontrivial divisors (so that there are no larger proper ideals). In $F[x]$, these are exactly the *irreducible* polynomials; in $\mathbb{Z}$, these are exactly the prime numbers.

As discussed above, since I is maximal (and R is commutative and unital), R/I is a field. Let us give a direct proof, one that gives us the technique to find inverses.

Suppose $b \notin I$; we want to show it is invertible in R/I. Now b is not a multiple of a and a is irreducible so we deduce that $\gcd(a,b) = 1$. By the extended Euclidean algorithm, we can write

$$1 = ra + tb$$

for some $r,t \in R$. Thus $tb + I = 1 + I$; this means $(t+I)(b+I) = 1 + I$ so $(b+I)$ is invertible. This holds for all $b \notin I$, in other words, every nonzero element of R/I is invertible. Thus R/I is a field. ■

■ **Example 8.8** The irreducibles in $\mathbb{Z}$ are the primes; so we recover the fact that $\mathbb{Z}/p\mathbb{Z}$ is a field for all primes p. ■

■ **Example 8.9** The polynomial x is irreducible, and $F[x]/\langle x \rangle \cong F$ is a field. ■

Now let us concentrate on the polynomial ring case. If F and E are two fields such that $F \subseteq E$, then E is called an *extension field* of F (and F is called a *subfield* of E).

The following is the key result that produces *all* finite fields.

Theorem 8.10 Let $R = F[x]$, for a field F and let m be an irreducible polynomial in $F[x]$ of degree n. Then the quotient ring

$$E = F[x]/\langle m \rangle$$

is an extension field of F, such that as a vector space over F, we have $\dim_F(E) = n$. If F is finite of order q, then E is of order q^n.

Proof. Recall by Lemma 6.27 that if $\deg(m) = n$ then we have

$$F[x]/\langle m \rangle = \{a_0 + a_1 x + \cdots a_{n-1}x^{n-1} \mid a_i \in F\}$$

(where on the right we perform operations mod $\langle m(x) \rangle$). This is a vector space over F of dimension n. If F is a finite field of order q, then this set has q^n elements.

Now if m is irreducible, then by Theorem 8.7

$$E = F[x]/\langle m \rangle$$

is a field. It contains F as a subfield (as the constant polynomials), so is an extension of F.

■

In the proof we noted a useful property: if E is an extension field of F then E is a vector space over F!

Definition 8.11 An extension field E of F such that $\dim_F(E) = n < \infty$ is called a *degree n extension* of the base field F (regardless of whether F is finite or not).

■ **Example 8.12** The polynomial x^2+1 is irreducible over $\mathbb{R}$; $\mathbb{R}[x]/\langle x^2+1\rangle$ is a field which is two-dimensional as a vector space over $\mathbb{R}$. Hence it must be $\mathbb{C}$.

We can write this out explicitly, to get to a more satisfactory reason why this field is $\mathbb{C}$: we have $F = \mathbb{R}$, $m(x) = x^2+1$, $I = \langle m\rangle$. Then

$$F[x]/I = \{a+bx \mid a,b \in \mathbb{R}\}$$

and we have the rule that $(a+bx)(c+dx) = ac+(ad+bc)x+bdx^2 \equiv ac-bd+(ad+bc)x \mod (x^2+1)$. This is now clearly the field $\mathbb{C}$, by the identification $a+bx \mapsto a+bi$ for all $a,b \in \mathbb{R}$. ■

In the last example, we saw that $F[x]/\langle m\rangle$ is a field where the symbol x is now interpreted as a (new!) root of the polynomial $m(x)$. This argument works to describe any field F.

■ **Example 8.13** Let $F = \mathbb{Z}_2$ and consider $p(x) = x^2+x+1$, which is an irreducible polynomial of degree 2. Then the set

$$\mathbb{F}_4 = F[x]/\langle p(x)\rangle = \{a_0+a_1x \mid a_0,a_1 \in \mathbb{Z}_2\}$$

should be a field, where we define addition as usual but do multiplication mod $p(x)$. In other words, since $p(x) \equiv 0 \mod \langle p(x)\rangle$, we have

$$x^2 = -x-1 = x+1 \in \mathbb{F}_4.$$

Thus, $\mathbb{F}_4 = \{0,1,x,1+x\}$ with addition mod 2 and multiplication mod $x^2 = x+1$.

Before writing out the addition and multiplication tables for $\mathbb{F}_4$, however, let's rename the variable, just as we did for $\mathbb{C}$. For example, let's replace x with α — we think of α as being a choice of root of the polynomial $p(x)$, just as we think of i as being one of the roots of $x^2+1=0$. Then addition in $\mathbb{F}_4$ satisfies

$+$	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

whereas multiplication in $\mathbb{F}_4$ is given by

$\times$	0	1	α	$1+\alpha$
0	0	0	0	0
1	0	1	α	$1+\alpha$
α	0	α	$1+\alpha$	1
$1+\alpha$	0	$1+\alpha$	1	α

We can see directly that this is a field (see 4 in Section 2.4); in particular, notice that it is NOT $\mathbb{Z}_4$. ■

R In fact, you can show directly that if E' is a field with 4 elements, then there is an isomorphism $E' \cong \mathbb{F}_4$. This is a general fact!

Summary: To construct a finite field with $q = p^n$ elements:

1. Find an irreducible polynomial $m \in \mathbb{Z}_p[x]$ of degree n;
2. Set $E = \mathbb{Z}_p[x]/\langle m\rangle$;
3. Choose a name for a root of $m(x)$, like α;
4. Identify $E = \{a_0 + a_1\alpha + \cdots a_{n-1}\alpha^{n-1} \mid a_i \in \mathbb{Z}_p, m(\alpha) = 0\}$.

These last two steps make a concrete field from the abstract construction. When $F = \mathbb{Q}$, then at step 3 it is instead: "choose a root $\alpha \in \mathbb{C}$ of $m(x)$" (and different choices can give different fields).

8.3 Exercises

1. Verify that the set $\{a + I \mid a \in F\} \subseteq F[x]/I$, for a maximal ideal I, equipped with the addition and multiplication in the quotient ring, is a field, and is isomorphic to F. (That is, we only take cosets of elements of F.)
2. Let E' be a field with 4 elements. Since it is a field, $0 \neq 1$ are two distinct elements of E'; so we can write $E' = \{0, 1, a, b\}$. Prove that (up to swapping the names of a and b) we must have $a^2 = b$ and $b = 1 + a$, and conclude that $E' \cong \mathbb{Z}_2[x]/\langle x^2 + x + 1\rangle$.
3. The polynomial $x^3 + x + 1$ is irreducible over $\mathbb{Z}_2$. Use it to define a finite field F with 8 elements and write down the multiplication table for F.
4. Now use the irreducible binary polynomial $x^3 + x^2 + 1$ to construct a finite field E with 8 elements instead. Find an isomorphism between F (of the preceding exercise) and E. (An isomorphism is a relabeling or permutation of elements such that the multiplication and addition tables become identical.)
5. Can you construct $\mathbb{F}_9$ using $x^2 + x + 1$ over $\mathbb{Z}_3$?
6. Construct $\mathbb{F}_9$ using $x^2 + x - 1$. Can you find a primitive element?
7. Write down all quadratic monic polynomials over $\mathbb{F}_4$. Identify an irreducible quadratic monic polynomial, and use it to construct a quadratic extension of $\mathbb{F}_4$, which you can call $\mathbb{F}_{16}$.
8. Find an irreducible quartic (degree 4) polynomial over $\mathbb{Z}_2$. Use it to construct a quartic extension of $\mathbb{Z}_2$ which has 16 elements. Show it is isomorphic to the field you constructed in the preceding exercise. With the help of your multiplication table, identify $\mathbb{F}_4$ as a subfield of F (that is, find an isomorphic copy of $\mathbb{F}_4$ inside F).
9. (Important Multiplication Theorem) Suppose $F \subseteq E \subseteq K$ are fields, such that E is a degree e extension over F and K is a degree k extension over E. Prove that K is a degree ek extension over F. *Hint: If $\{e_i\}$ is a basis of E over F, and $\{k_j\}$ is a basis of K over E, prove directly that $\{e_ik_j\}$ is a basis for E over F. Alternately: prove this just in the finite field case.*

8.4 Representing elements of a finite field in two ways

Consider $F = \mathbb{C}$. We have two representations of elements of F:

- cartesian form (where an element is written as $a+bi$ for some $a,b \in \mathbb{R}$) and
- polar form (where an element is written as $re^{i\theta}$ for some $r,\theta \in \mathbb{R}$, $r \geq 0$).

One reason why we use two different forms is the following pair of observations:

- Adding complex numbers in Cartesian form is easy; adding numbers in polar form is hard.
- Multiplying complex numbers in polar form is easy; multiplying them in cartesian form takes more work.

Now consider F a finite field, constructed as $\mathbb{Z}_p[x]/\langle m \rangle$ as in the previous section. Suppose $\deg(m) = n$. We write an element of F in *standard form* as

$$b_0 + b_1\alpha + \cdots + b_{n-1}\alpha^{n-1}$$

for some $b_i \in \mathbb{Z}_p$. Then addition of elements of F is easy to write down, but multiplication takes extra work. Our question is: can we find a different way of representing the elements of F, such that multiplication is easy to work out in this new form?

We consider this in some examples.

■ **Example 8.14** Consider $\mathbb{F}_4 = \{0,1,\alpha,1+\alpha \mid \alpha^2 = \alpha+1\}$ as in Example 8.13. Since

$$\alpha^3 = \alpha^2\alpha = (1+\alpha)(\alpha) = \alpha + \alpha^2 = \alpha + (1+\alpha) = 1,$$

this is the set

$$\mathbb{F}_4 = \{0,1,\alpha,\alpha^2 \mid \alpha^3 = 1\}.$$

Multiplication in $\mathbb{F}_4$ is thus given by the rules: $0 \times t = 0$ for all t; and $\alpha^k\alpha^\ell = \alpha^{k+\ell \mod 3}$. ■

■ **Example 8.15** Consider $\mathbb{F}_8 = \mathbb{Z}_2[x]/\langle x^3+x+1 \rangle$. Then in standard form we have

$$\mathbb{F}_8 = \{a_0 + a_1\beta + a_2\beta^2 \mid a_i \in \mathbb{Z}_2, \beta^3 = \beta+1\}.$$

We calculate:

- $\beta^3 = 1+\beta$
- $\beta^4 = \beta+\beta^2$
- $\beta^5 = \beta^2(1+\beta) = \beta^2+\beta^3 = 1+\beta+\beta^2$
- $\beta^6 = (\beta+1)^2 = 1+\beta^2$
- $\beta^7 = \beta(1+\beta^2) = \beta+(1+\beta) = 1$

so in fact

$$\mathbb{F}_8 = \{0,1,\beta,\beta^2,\cdots,\beta^6\}$$

which is rather convenient for working out multiplication in $\mathbb{F}_8$. ■

Definition 8.16 Let F be a finite field of order q. An element $\alpha \in F$ with the property that

$$F^\times = \{1,\alpha,\cdots,\alpha^{q-2} \mid \alpha^{q-1} = 1\}$$

is called a *primitive element* of F.

Since $F^\times$ is a multiplicative group of order $q-1$, Lagrange's theorem promises us that each element of $F^\times$ has order *dividing* $q-1$. The existence of a primitive element of F is a much stronger assertion: it can happen if and only if $F^\times$ is a *cyclic group*. (See Corollary **??**, coming up.)

■ **Example 8.17** Consider $\mathbb{F}_9 = \{a_0 + a_1\gamma \mid \gamma^2 + 1 = 0, a_i \in \mathbb{Z}_3\}$. Then

$$\gamma^2 = 2, \gamma^3 = 2\gamma, \gamma^4 = 1$$

so γ has order 4. Thus the order of γ divides $q-1=8$, but γ is not a primitive element. ■

8.5 Application: minimal polynomials

We mentioned earlier that when we write β (or α, or γ) in place of x in our definition of a field, like

$$\mathbb{F}_8 = \{a_0 + a_1\beta + a_2\beta^2 \mid a_i \in \mathbb{Z}_2, \beta^3 + \beta + 1 = 0\},$$

then we are thinking of β as being one root of the polynomial $x^3 + x + 1$. This is reasonable, because in $\mathbb{F}_8$, we have that $\beta^3 + \beta + 1 = 0$, so it really is a root.

We might ask:

- *Are there other roots of this polynomial in* $\mathbb{F}_8$*?*
  For example, in $\mathbb{C}$ there are two roots of $x^2 + 1$, namely i and $-i$.
- *Do all elements of* $\mathbb{F}_8$ *satisfy this polynomial?*
  Wait: that's an obvious NO. If a polynomial is cubic, it has at most 3 roots, and $\mathbb{F}_8$ has 8 elements.
- In $\mathbb{C}$, we know that every complex (non-real) number is the root of a quadratic irreducible polynomial.
  Is it true that every element of $\mathbb{F}_8$ *is the root of some irreducible polynomial? And if so, how do we find it and what does it tell us?*

Let's work this all out in our example. Our objective is:

For each $a \in \mathbb{F}_8$, find an irreducible polynomial of which it is a root.

Recall that we had worked out the additive-multiplicative dictionary:

additive form	multiplicative form
0	-
1	$1 = \beta^7$
β	β
β^2	β^2
$1+\beta$	β^3
$\beta+\beta^2$	β^4
$1+\beta+\beta^2$	β^5
$1+\beta^2$	β^6

Consider the element β^2**.** We calculate

$$(\beta^2)^3 + \beta^2 + 1 = (1+\beta^2) + \beta^2 + 1 = 0$$

so β^2 is another root of $x^3 + x + 1$.

Now consider β^3**.** We calculate

$$(\beta^3)^3 + \beta^3 + 1 = \beta^2 + (1+\beta) + 1 \neq 0$$

so β^3 is NOT a root of x^3+x+1. How can we find a polynomial that β^3 satisfies?

One approach: think of a polynomial equation in some element γ as being a dependence relation on the set $\{1,\gamma,\gamma^2,\cdots\}$. Since $\mathbb{F}_8$ is a 3-dimensional vector space over $\mathbb{Z}_2$, the set $\{1,\gamma,\gamma^2,\gamma^3\}$ must be linearly dependent, meaning, there must exist a polynomial of degree at most 3 of which γ is a root.

Let's consider $\gamma=\beta^3$, again. Using our relations we have

$$\{1,\gamma,\gamma^2,\gamma^3\}=\{1,\beta^3,\beta^6,\beta^9\}=\{1,1+\beta,1+\beta^2,\beta^2\}.$$

We see in this case that the sets $\{1\}$, $\{1,\gamma\}$, and $\{1,\gamma,\gamma^2\}$ are all linearly independent; but the set $\{1,\gamma,\gamma^2,\gamma^3\}$ is dependent, with dependence relation:

$$1+\gamma^2+\gamma^3=0 \quad \text{that is,} \quad (1)+(1+\beta^2)+(\beta^2)=0.$$

So β^3 is a root of x^3+x^2+1, and this is the (monic) polynomial of minimal degree which β^3 satisfies. We therefore call it the *minimal polynomial* of β^3.

R Notice that x^3+x^2+1 is another irreducible polynomial of degree 3 which we could have used to construct a field with 8 elements (as in some of the exercises above).

Now take $\gamma=\beta^4$. Then

$$\{1,\gamma,\gamma^2,\gamma^3\}=\{1,\beta^4,\beta^8=\beta,\beta^{12}=\beta^5\}=\{1,\beta+\beta^2,\beta,1+\beta+\beta^2\}$$

Again, we find by looking at these elements that the minimal polynomial of β^4 is x^3+x+1.

Exercise : β^5 and β^6. By either a similar method, or by plugging into the existing list of candidates, we work out that β^5 and β^6 both have minimal polynomial x^3+x^2+1.

While we're at it: the minimal polynomial of $\gamma=1$ is $x-1$; and the minimal polynomial of $\gamma=0$ is x.

Conclusion:

- Each element of $\mathbb{F}_8$ has a minimal polynomial from the following list:

$$x, x-1, x^3+x+1, x^3+x^2+1.$$

- Each of these polynomials is irreducible over $\mathbb{Z}_2$, but factor completely into linear factors over $\mathbb{F}_8$.
- The 8 elements of $\mathbb{F}_8$ are precisely all the roots of these 4 polynomials.
- When we multiply them all together we get

$$x(x-1)(x^3+x+1)(x^3+x^2+1)=x(x^7-1)=x^8-x.$$

We'll see in the next section that this is a general fact.

Let us summarize some important phenomena we have identified here.

Lemma 8.18 Let E be a finite extension of F and $\gamma \in E$. Then any element $\gamma \in E$ is the root of some polynomial with coefficients in F. There is moreover a unique monic polynomial $m(x) \in F[x]$ of minimal degree such that $m(\gamma) = 0$.

We leave the proof as an exercise. One needs to justify both the existence, and the assertion of uniqueness. Note that existence requires the hypothesis of E being a finite extension of F.

Definition 8.19 The *minimal polynomial* of γ over F is the unique monic polynomial of minimal degree $m(x) \in F[x]$ such that $m(\gamma) = 0$.

8.6 Exercises

1. Let $\mathbb{F}_9$ be defined by the irreducible polynomial x^2+1 over $\mathbb{Z}_3$. Find all primitive elements of $\mathbb{F}_9$.
2. Prove that if F has one primitive element, then it has $\varphi(q-1)$ primitive elements, where φ is the Euler phi function, that is, $\varphi(n) = |\mathbb{Z}/n\mathbb{Z}^\times|$.
3. Prove that (in the setup of the definition of the minimal polynomial) that if $m(x), n(x) \in F[x]$ are two monic polynomial of (the same) minimal degree for which $m(\gamma) = n(\gamma) = 0$, then $m = n$, that is, the polynomials are equal, and the minimal polynomials is uniquely characterized in this way.
4. Prove that if $\gamma \in E$ has minimal polynomial $m(x) \in F[x]$, and $f(x) \in F[x]$ is any other polynomial of which γ is a root, then $m(x)$ divides $f(x)$.
5. Give an example of an element in $\mathbb{R}$ which is not the root of a polynomial with coefficients in $\mathbb{Q}$ (that is, which is not *algebraic* over $\mathbb{Q}$). Why does our argument in the main example of Section not apply?
6. Give the minimal polynomial for $\gamma = \sqrt{2}$ over $\mathbb{Q}$.
7. Suppose F is a finite field containing $\mathbb{Z}_p$, and let $\gamma \in F^\times$.
   (a) Argue that there exists some $n \geq 1$ such that the set $\{1, \gamma, \cdots, \gamma^{n+1}\}$ is linearly dependent over $\mathbb{Z}_p$.
   (b) Let n be such a value. Argue that there is a polynomial of degree at most n in $\mathbb{Z}_p[x]$ of which γ is a root.
   (c) Prove that if n is the least such value, then this polynomial must be irreducible, and hence, by dividing by its leading coefficient to make it monic, it gives the minimal polynomial of γ over $\mathbb{Z}_p$.
   (d) Prove that if $\gamma \in \mathbb{Z}_p$ then $\{1, \gamma\}$ is linearly dependent over $\mathbb{Z}_p$. Give the minimal polynomial of γ.
8. In this question, you will determine the key properties of the field $GF(9)$ and record them in a table. (Hang on to this table.)
   (a) Construct the field $GF(9)$.
   (b) Find a primitive element.
   (c) Create a table where you write out the additive form of each power of your primitive element.
   (d) Add a column where you write down the order of each element in the multiplicative group of the field.
   (e) Add a row for 0 (which is not a power of your primitive element, of course).

(f) Now calculate the minimal polynomial of each element of $GF(9)$ and record that in a fourth column. Use your multiplicative-additive dictionary to do this easily.
(g) Finally, verify that the least common multiple of all the minimal polynomials is equal to $x^9 - x$.

8.7 Main theorems about finite fields

The detailed example of the previous section revealed many interesting facts about $\mathbb{F}_8$ that are true in general. Let's round out our discussion of finite fields with the statements of some key theorems. The proofs of some results are included but require mathematical background above what has been assumed thus far in these notes.

Let E be a finite field of order q. Then the set $E^\times$ is a multiplicative group of order $q-1$, so by Lagrange's theorem[1], each element of $E^\times$ satisfies

$$a^{q-1} = 1.$$

To include $a = 0$, multiply this by a. So every element of E satisfies $a^q = a$; in fact, E is precisely the set of solutions in E to the equation

$$x^q - x = 0.$$

R This is an extremely special situation, for finite fields. For infinite fields, one instead introduces the notion of a *splitting field* for a polynomial $m(x) \in F[x]$, which is an extension field E of F of minimal degree such that $m(x)$ factors completely into linear factors in $E[x]$ (so that E contains all the roots of m). Since F and E are infinite, it is impossible for E to be the set of all roots of one polynomial!

We can now state the key theorem of finite fields. We attribute it to Évariste Galois in honour of his contributions to the modern understanding of these fields in his 1830 paper [Gal30], but the result does not appear there. It does appear in the first chapter of an important treatise of Camille Jordan in 1870 [Jor70].

Theorem 8.20 — Galois 1830. Let p be a prime and $n \geq 1$. Then there exists a field of order p^n, and it is unique up to isomorphism.

We sometimes call the field of order p^n the *Galois field* of order p^n, denoted $GF(p^n)$ or $\mathbb{F}_{p^n}$.

Idea of proof. We want to define E as the set of all solutions to $x^{p^n} - x$. One tricky step is to prove that such a polynomial couldn't have repeated roots. Another is to make sure this is a field. For this latter problem, we could start by postulating the existence of an algebraic closure K of $\mathbb{Z}_p$, in which case $E \subseteq K$ and you show it's a subfield; but this is cheating, since the usual way we define K is as $\cup_{n\geq 1}\mathbb{F}_{p^n}$.

The longer route is to prove the existence of splitting fields, and show that if E' is a splitting field for $x^{p^n} - x$ then the subset E consisting of just the roots of this polynomial is in fact already a field. ■

[1] Lagrange's theorem asserts that if G is a group with k elements, then any element $g \in G$ satisfies $g^k = 1$; therefore, its order n, which is the minimal power such that $g^n = 1$, must divide k.

We deduce several consequences that make finite fields quite pleasant to work with.

Corollary 8.21 Let $E = \mathbb{F}_{q^n}$ be a degree n extension of a finite field $F = \mathbb{F}_q$ of order q. Then:

1. The minimal polynomial of any $a \in E$ over F must be an irreducible factor of $x^{q^n} - x$. In particular, $a^{q^n} - a = 0$ for all $a \in E$.
2. Every irreducible polynomial of degree n over F divides $x^{q^n} - x$ in $F[x]$.

Proof. (1) Let $g(x) = x^{q^n} - x$ and let $a \in E$. Then $g(a) = 0$. Let m be the minimum polynomial of a; then since $m(a) = 0$ it follows that if $d(x) = \gcd(m, g)$, we have $d(a) = 0$. Since m is irreducible, we deduce that $d = m$ and so m divides g.

(2) Given an irreducible polynomial $p(x)$ of degree n, construct a field of order q^n using $p(x)$. Then any root of $p(x)$, including $x = \alpha$, has minimal polynomial $p(x)$. By (1), we conclude that $p(x)$ divides $x^{q^n} - x$. ■

This settles many of our open questions from our examples in previous sections; let us also settle the last, namely the existence of primitive elements.

Theorem 8.22 If F is a finite field then $F^\times$ is cyclic, so has a primitive element.

We defer the proof of this statement to the following section, and instead derive the immediate consequences.

Corollary 8.23 Let F be a finite field. If E is a field extension of F of degree n then there exists an irreducible polynomial of degree n in $F[x]$.

Proof. Since E is a field, we have by the theorem that $E^\times$ is a cyclic group. Let $\alpha \in E^\times$ be a generator, and consider its minimal polynomial $m(x) \in F[x]$ over F. If $\deg(m) = k < n$, then α^k is a linear combination of $\{1, \alpha, \cdots, \alpha^{k-1}\}$. It follows that $\alpha^\ell \in \operatorname{span}_F\{1, \alpha, \cdots, \alpha^{k-1}\}$ for all $\ell \geq 1$. But there are only $|F|^k$ elements in this span whereas the cyclic group generated by α has order $|F|^n - 1$, a contradiction. ■

This corollary tells us, for example, that *every* finite field of characteristic p can be realized as $\mathbb{Z}_p[x]/\langle m \rangle$ for some irreducible $m \in \mathbb{Z}_p[x]$.

8.8 Proof of the primitive element theorem

We prove something a bit more general than Theorem 8.22. Recall that E being a splitting field of a polynomial $f \in F[x]$ means that E contains all the roots of f.

Theorem 8.24 Suppose E is a splitting field of $x^n - 1$. If $char(E) = p$, assume furthermore that $\gcd(n, p) = 1$. Then the set μ_n of all nth roots of unity in E is a cyclic group.

To prove this theorem, we use an unusual characterisation of cyclic groups.

Lemma 8.25 A group G of order n is cyclic if and only if there exists exactly one subgroup of order d, for any d dividing n.

"Only if" direction of the proof. Throughout this proof, let us write $\langle h\rangle$ for the subgroup of G generated by h; this is the set $\{1,h,h^2,\cdots,h^{k-1} \mid h^k=1\}$. If k is the least integer such that $h^k=1$, then (exercise), this cyclic subgroup has order k.

Suppose that $G=\langle g\rangle$ is a cyclic group of order n (so $g^n=1$, and this is the least positive power for which this holds). We begin with a few facts, which you can prove as an exercise:

- For any two $g^k, g^\ell \in G$, we have that $g^k, g^\ell \in \langle g^d\rangle$ where $d=\gcd(k,\ell)$, and any group containing g^k and g^ℓ contains g^d.
- Every subgroup of G is cyclic.
- The cyclic subgroup of G generated by g^k is also generated by g^d, where $d=\gcd(k,n)$, and this group has size n/d.
- There is exactly one cyclic subgroup of order d for each d which is a divisor of n, and it is generated by (for example) $g^{n/d}$.

These facts together give one implication of the lemma. ■

Before proving the reverse implication, we need another lemma.

Lemma 8.26 Let φ denote the Euler totient function. Then the number of elements of order exactly n in a cyclic group of order n is $\varphi(n)$. Furthermore, we have

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

Proof. In the cyclic group of order n generated by g, the element g^k is another generator if and only if $\gcd(k,n)=1$ (because otherwise, $\langle g^k\rangle=\langle g^d\rangle\neq G$). What this means is: g^k has order exactly n iff $\gcd(k,n)=1$. So the number of elements of order n in G is $\varphi(n)$, the Euler phi function.

Since all subgroups of G are cyclic, we deduce that the number of generators of a cyclic subgroup of order d is exactly $\varphi(d)$; this means the number of elements of order exactly d in G is $\varphi(d)$, for any divisor d of n.

Now count the number of elements of any given order in G. These orders must divide n. Let d divide n. Each element of order d generates a subgroup of G of order d; but there is only one such subgroup, and it therefore contains all the elements of order d. How many? $\varphi(d)$. In this way we account for all the n elements of G, and conclude the surprisingly useful formula:

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

which has nothing to do with cyclic groups, but is a nice formula in number theory. ■

“If" direction of the proof. Now let us prove the inverse implication of Lemma 8.25. That is, suppose G has this unique subgroup property. Proceeding as above, let h be an element of order d (which must divide n). Then the cyclic subgroup generated by h has order d, and is the unique subgroup of order d. We conclude (by our discussion about the number of generators of any cyclic group) that there are exactly $\varphi(d)$ elements in G of order d. So: for each d dividing n, including n itself, there are either 0 or $\varphi(d)$ elements of order exactly d. But the total number of elements must be $n = |G|$. So by the above useful equality, every d must occur as the order of some element of G, and so in particular G must have an element of order n, and so be cyclic. ■

Proof of Theorem 8.24. By Lemma 8.25, it therefore suffices to show that for each divisor d of n there exists at most one subgroup of μ_n of order d. In a subgroup H of order d, by Lagrange's theorem, every element satisfies $x^d = 1$. Since we are working over a field, this says every element of H is a root of $x^d - 1$. Since $x^n - 1 = (x^d - 1)(x^{n-d} + x^{n-2d} + \cdots + 1)$, every root of $x^d - 1$ is also a root of $x^n - 1$, so lies in μ_n.

In particular, $x^d - 1$ can have no repeated roots; this is guaranteed by the hypothesis that $\gcd(n, p) = 1$ (exercise). Therefore if we had two subgroups of order d, we'd have more than d solutions to the equation $x^d - 1 = 0$, which is impossible. ■

Now let E be a finite field of order $q = p^\ell$, and set $n = q - 1$. Then we have $\gcd(n, p) = 1$, so by Theorem 8.24, the set of nth roots of unity is a cyclic group. But by Corollary 8.21, the nth roots of unity are precisely the elements of $E^\times$; thus Theorem 8.22 follows.

Given that $E^\times = \mu_{q-1}$, the following definition is natural.

Definition 8.27 An element β of order n in an extension field of $\mathbb{Z}_p$, where $\gcd(n, p) = 1$, is called a *primitive nth root of unity*.

8.9 Exercises

1. Prove all the statements about cyclic groups stated at the beginning of the lemma. (See [Nic12, Chapter 2.4].)
2. Prove that the generators of a cyclic group $G = \{1, g, \cdots, g^{n-1} \mid g^n = 1\}$ are exactly $\{g^k \mid \gcd(k, n) = 1\}$.
3. Use the formal derivative of a polynomial to prove that $x^n - 1$ has no repeated roots in any extension of $\mathbb{Z}_p$ if and only if $\gcd(n, p) = 1$. (See [Nic12, Chapter 6.4 Theorem 2].) *Hint: argue that $f \mapsto f'$ is a linear map on $F[x]$ for any F. Then observe that from Calculus we know that if $f(x) = (x-a)^2 g(x)$ then $f'(x) = (x-a)(2g(x) + (x-a)g'(x))$ whereas if $f(x) = (x-a)g(x)$ (with $g(a) \neq 0$) then $f'(x) = g(x) + (x-a)g'(x)$, to argue the result.*
4. Show that if n divides $q - 1$, where q is a prime power, then $\mu_n \subset GF(q)$. Infer that $x^n - 1$ factors into irreducible factors in any field $GF(q)$ such that $n|(q-1)$.

9. BCH codes and Reed-Solomon Codes

We previously saw how to construct and efficient decode Hamming codes (which all have $d_{min} = 1$). In this chapter, we continue our work with cyclic codes, applying the insights we gained into the factorization of $x^n - 1$ in our study of finite fields. End result: we'll be able to *design* codes of *any* minimum distance, and we'll also gain an efficient decoding algorithm that makes them quite practical. In consequence, even though BCH codes and Reed-Solomon codes are just the cyclic codes of Chapter 7, they have earned their own name!

9.1 The Vandermonde determinant

Before we tackle BCH codes, there's a common and extremely useful tool we'll need, so we'll state it as a proposition: it's called the *Vandemonde determinant*.

Proposition 9.1 Let $s \geq 1$ and let $\{x_1, \cdots, x_s\}$ be a set of s commuting symbols. Let $V_s = V_s(x_1, \cdots, x_s)$ denote the $s \times s$ matrix

$$V_s = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_s \\ x_1^2 & x_2^2 & \cdots & x_s^2 \\ \vdots & \vdots & & \vdots \\ x_1^{s-1} & x_2^{s-1} & \cdots & x_s^{s-1} \end{bmatrix}.$$

Then we have

$$\det(V_s) = \prod_{1 \leq i < k \leq s} (x_k - x_i).$$

The Vandermonde determinant is named after Alexandre–Théophile Vandermonde (1735–96). While he is generally regarded as the “founder of the theory of determinants", and he proved the techniques we will use in the proof, ironically he did not actually produce the result that bears his name [OR01].

Proof. We proceed by induction. We know that $\det(V_1) = 1$. Suppose that $s \geq 2$ and we know that

$$\det(V_{s-1}(x_2, \cdots, x_s) = \prod_{2 \leq i < k \leq s} (x_k - x_i).$$

We now row reduce V_s, keeping track of our elementary row operations so that we can relate $\det(V_s(x_1, \cdots, x_s))$ to $\det(V_{s-1}(x_2, \cdots, x_s))$.

Recall that if we perform a row operation of the form $c \times \mathrm{Row}(i) + \mathrm{Row}(i+1) \to \mathrm{Row}(i+1)$, then this does not change the value of the determinant. Also, if we factor a scalar z from a row or column of A, then the determinant of the resulting matrix is $z\det(A)$. (See the exercises.)

To compute

$$\det V_s = \det \begin{bmatrix} 1 & 1 & \cdots \\ x_1 & x_2 & \cdots \\ x_1^2 & x_2^2 & \cdots \\ \vdots & \vdots & \\ x_1^{s-1} & x_2^{s-1} & \cdots \end{bmatrix}$$

we cleverly clear out the first column by adding $-x_1 \times \mathrm{Row}(s-1)$ to $\mathrm{Row}(s)$, then $-x_1 \times \mathrm{Row}(s-2)$ to $\mathrm{Row}(s-1)$, and so on, to deduce

$$\det(V_s) = \det \begin{bmatrix} 1 & 1 & 1 & \cdots \\ 0 & x_2 - x_1 & x_3 - x_1 & \cdots \\ 0 & x_2(x_2 - x_1) & x_3(x_3 - x_1) & \cdots \\ \vdots & \vdots & & \\ 0 & x_2^{s-2}(x_2 - x_1) & x_3^{s-2}(x_3 - x_1) & \cdots \end{bmatrix}.$$

Now expand along the first column to deduce that the above determinant is

$$= 1 \times \det \begin{bmatrix} x_2 - x_1 & x_3 - x_1 & \cdots \\ x_2(x_2 - x_1) & x_3(x_3 - x_1) & \cdots \\ \vdots & \vdots & \\ x_2^{s-2}(x_2 - x_1) & x_3^{s-2}(x_3 - x_1) & \cdots \end{bmatrix}.$$

Finally, take out the common factor in each column to get

$$\begin{aligned} \det(V_s) &= (x_2 - x_1)(x_3 - x_1) \cdots (x_s - x_1) \det \begin{bmatrix} 1 & 1 & \cdots \\ x_2 & x_3 & \cdots \\ x_2^2 & x_3^2 & \cdots \\ \vdots & \vdots & \\ x_2^{s-2} & x_3^{s-2} & \cdots \end{bmatrix} \\ &= (x_2 - x_1)(x_3 - x_1) \cdots (x_s - x_1) \det(V_{s-1}(x_2, \cdots, x_s) \\ &= \prod_{1 = i < k \leq s} (x_k - x_i) \cdot \prod_{2 \leq i < k \leq s} (x_k - x_i) \\ &= \prod_{1 \leq i < k \leq s} (x_k - x_i). \end{aligned}$$

as required. ■

One of the key applications is the following. Suppose we want to find a polynomial of degree $s-1$ whose graph interpolates some points (over some field F):

$$(x_1,y_1),(x_2,y_2),\cdots,(x_s,y_s)\in F^2.$$

That is, we're trying to find a polynomial

$$f(x)=c_0+c_1x+\cdots+c_{s-1}x^{s-1}\in F[x]$$

such that

$$\forall 1\le i\le s,\quad f(x_i)=y_i.$$

Let c be the (column) vector of the unknown coefficients $(c_0,c_1,\cdots,c_{s-1})$ and set $y=(y_1,\cdots,y_s)$. Letting $A=V_s^T(x_1,\cdots,x_s)$, our problem is to solve the matrix equation

$$Ac=y$$

for c. Since $\det(A)=\det(V_s^T(x_1,\cdots,x_s))\neq 0$ if and only if all of the values x_i are distinct, we conclude that this system has a unique solution if and only if the x_i are distinct. That is, we can always uniquely interpolate s values (satisfying this condition) with a polynomial of degree $s-1$ (independent of the field we are working over).

9.2 The BCH theorem

In this chapter, we assume that $\gcd(n,p)=1$, where n is the length of our code and p is the characteristic of our field F. Then we can look for $q=p^m$, a power of p, such that $n|(q-1)$. (Exercise: such a q always exists.) When this happens, Theorem 8.22 and Lemma 8.25 together tell us that we'll have $\mu_n\subset\mathbb{F}_q^\times$.

In particular, $n|(q-1)$ implies that x^n-1 factors into linear factors over $\mathbb{F}_q$. Since μ_n is cyclic, there is some $\gamma\in\mathbb{F}_q$ that generates μ_n. (In general this is some power of a primitive element of $\mathbb{F}_q$ unless $\mu_n=\mathbb{F}_q$.) Therefore, the roots of x^n-1 are $\mu_n=\{1,\gamma,\gamma^2,\cdots,\gamma^{n-1}\}$ so x^n-1 factors as

$$x^n-1=(x-1)(x-\gamma)(x-\gamma^2)\cdots(x-\gamma^{n-1}).$$

Now suppose that $g(x)$ divides x^n-1. Then we can factor

$$g(x)=\prod_{j\in S}(x-\gamma^j)$$

for some subset $S\subseteq\{0,1,\cdots,n-1\}$. This observation led to the following remarkable theorem, due to Hocquenghem [Hoc59] and, independently, by Bose and Ray-Chaudhuri [BRC60].

Theorem 9.2 Let $C = \langle g(x) \rangle$ be a cyclic code of length n. Let γ be a primitive nth root of unity over $\mathbb{Z}_p$. If t consecutive powers of γ are roots of $g(x)$, that is, if there exists an ℓ such that

$$g(\gamma^\ell) = g(\gamma^{\ell+1}) = \cdots = g(\gamma^{\ell+t-1}) = 0,$$

then $d_{min}(C) > t$.

Proof. Let $E = \mathbb{F}_q$ be an extension field of $\mathbb{Z}_p$ which contains μ_n, and let $\gamma \in \mu_n$ be a primitive nth root of unity. We want to show that under the given hypothesis, the weight of any nonzero vector in C is greater than t.

Let $c(x) \in C \setminus \{0\}$; then $c(x) \in \langle g(x) \rangle$ so there exists some $q(x)$ such that $c(x) = q(x)g(x) \mod \langle x^n - 1 \rangle$. More precisely, there exists some $r(x) \in \mathbb{Z}_p[x]$ such that

$$c(x) = q(x)g(x) + r(x)(x^n - 1).$$

Since any power of γ is a root of $x^n - 1$, and the elements $\{\gamma^\ell, \gamma^{\ell+1}, \cdots, \gamma^{\ell+t-1}\}$ are roots of g, we conclude that $\{\gamma^\ell, \gamma^{\ell+1}, \cdots, \gamma^{\ell+t-1}\}$ are each roots of $c(x)$.

Now suppose to the contrary that $wt(c) = s \le t$. Then we could identify exactly s nonzero coefficients of $c(x)$, say $c_{j_1}, c_{j_2}, \cdots, c_{j_s}$. Then the first s equations $c(\gamma^\ell) = 0, \cdots, c(\gamma^{\ell+s-1}) = 0$ give rise to a system of s homogeneous linear equations in the s variables $c_{j_1}, c_{j_2}, \cdots, c_{j_s}$:

$$\begin{aligned} c_{j_1}(\gamma^\ell)^{j_1} + \cdots + c_{j_s}(\gamma^\ell)^{j_s} &= 0 \\ c_{j_1}(\gamma^{\ell+1})^{j_1} + \cdots + c_{j_s}(\gamma^{\ell+1})^{j_s} &= 0 \\ \vdots &= \vdots \\ c_{j_1}(\gamma^{\ell+s-1})^{j_1} + \cdots + c_{j_s}(\gamma^{\ell+s-1})^{j_s} &= 0. \end{aligned}$$

The square matrix of coefficients is

$$B = \begin{bmatrix} \gamma^{j_1\ell} & \gamma^{j_2\ell} & \cdots & \gamma^{j_s\ell} \\ \gamma^{j_1(\ell+1)} & \gamma^{j_2(\ell+1)} & \cdots & \gamma^{j_s(\ell+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma^{j_1(\ell+s-1)} & \gamma^{j_2(\ell+s-1)} & \cdots & \gamma^{j_s(\ell+s-1)} \end{bmatrix}.$$

Since the above system has a nontrivial solution (namely, the one coming from $c(x)$), this matrix is not invertible. Thus its determinant is zero. Let's calculate its determinant.

$$\det(B) = \gamma^{j_1\ell}\gamma^{j_2\ell}\cdots\gamma^{j_s\ell} \det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \gamma^{j_1} & \gamma^{j_2} & \cdots & \gamma^{j_s} \\ \gamma^{2j_1} & \gamma^{2j_2} & \cdots & \gamma^{2j_s} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma^{(s-1)j_1} & \gamma^{(s-1)j_2} & \cdots & \gamma^{(s-1)j_s} \end{bmatrix}.$$

This latter determinant is exactly the Vandermonde determinant of Proposition 9.1, with $x_i = \gamma^{j_i}$ for $i \in \{1, \ldots, s\}$. Therefore, we have

$$\det(B) = \gamma^{j_1\ell}\gamma^{j_2\ell}\cdots\gamma^{j_s\ell}\prod_{i<k}(\gamma^{j_k} - \gamma^{j_i}).$$

But since all of our γ^{j_i} are distinct (by primitivity of γ), and nonzero (since they are roots of unity), this determinant *cannot* be zero, which is a contradiction.

Hence: there does not exist a nonzero codeword $c(x) \in C$ of weight $s \leq t$, whence $d_{min} > t$. ■

This is marvelous! It is a huge and strange result. It suggests we can — and indeed it is possible to — construct cyclic codes with arbitrarily high values of d_{min}.

Reality check: If we fix n, however, note that the theorem asks us to find $g(x)$ with t consecutive powers of γ as a root, and it is possible that the only divisor of $x^n - 1$ which has this property, for a given value of t, is $g(x) = x^n - 1$, which corresponds to the zero code. But if we're willing to go to large values of n, this method gives us incredibly good error-correcting codes.

Our next question: this is a wonderful theory, but how do we find γ, and how do we compute with these things?

9.3 Designed distance codes or BCH codes

To apply Theorem 9.2, we need to find a primitive nth root of unity, γ. Where do we start?

We are looking for an element γ such that γ has order n. Then in particular γ is a root of $x^n - 1$, which implies that the minimal polynomial $m(x)$ (of γ over $\mathbb{Z}_p$) must divide $x^n - 1$ (Exercise 4 of Section 8.6). Moreover, if $m(x)$ were to divide $x^k - 1$, for any $k < n$, then we'd have $\gamma^k = 1$, contradicting that n is the *smallest* power of γ which equals 1.

This line of reasoning tells us where to look for elements like γ. Namely, choose an irreducible factor $m(x)$ of $x^n - 1$ such that $m(x)$ does not divide $x^k - 1$ for any $k < n$. (The existence of such a factor depends on the hypothesis that $\gcd(n, p) = 1$, although we do not prove it here.)

Suppose $\deg(m(x)) = m$. Construct the field $E = \mathbb{F}_{p^m}$ as $E = \mathbb{Z}_p[x]/\langle m(x) \rangle$, and let γ be the image of x in E. Then by construction, $m(\gamma) = 0$, so we conclude both that $\gamma^n = 1$ (since $m(x)$ divides $x^n - 1$) and that $\gamma^k \neq 1$ (since $m(x)$ does not divide $x^k - 1$) for any $0 < k < n$.

■ **Example 9.3** Say $n = 5$ and $p = 2$. To find $\gamma \in \mu_5$, a primitive 5th root of unity, we first factor

$$x^5 - 1 = (x-1)(x^4 + x^3 + x^2 + x + 1).$$

The first factor clearly divides $x - 1$, so doesn't work. The second one clearly cannot factor any $x^r - 1$, for $r \leq 4$, so meets our criterion. Set $m(x) = x^4 + x^3 + x^2 + x + 1$. Then $F[x]/\langle m(x) \rangle$ is a field with $2^4 = 16$ elements. So there exists an element of $\mathbb{F}_{16}$ of order exactly 5, which we could call γ. It isn't hard to see that γ, γ^2, γ^3 and γ^4 are all roots of $m(x)$, so if we set $g(x) = m(x)$ we have a polynomial that satisfies Theorem 9.2 with $t = 4$. The corresponding code is $C = \{00000, 11111\}$, which has $d_{min} = 5 > 4$, as expected. ■

That was a lot of work. It's more efficient to turn this around: let's first start with a given field $\mathbb{F}_q$ such that $n | (q-1)$ and identify the order of all of the elements of $\mathbb{F}_q^\times$ as well as their minimal polynomials. Then we can select the generator of our codes from the data at hand.

multiplicative form	additive form	minimal polynomial	order of element
-	0	x	-
1	1	$x-1$	1
α	α	$p(x)$	15
α^2	α^2	$p(x)$	15
α^3	α^3	$r(x)=x^4+x^3+x^2+x+1$	5
α^4	$1+\alpha$	$p(x)$	15
α^5	$\alpha+\alpha^2$	$t(x)=x^2+x+1$	3
α^6	$\alpha^2+\alpha^3$	$r(x)$	5
α^7	$1+\alpha+\alpha^3$	$s(x)=x^4+x^3+1$	15
α^8	$1+\alpha^2$	$p(x)$	15
α^9	$\alpha+\alpha^3$	$r(x)$	5
α^{10}	$1+\alpha+\alpha^2$	$t(x)$	3
α^{11}	$\alpha+\alpha^2+\alpha^3$	$s(x)$	15
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	$r(x)$	5
α^{13}	$1+\alpha^2+\alpha^3$	$s(x)$	15
α^{14}	$1+\alpha^3$	$s(x)$	15

Table 9.1: The field $\mathbb{F}_{16}$, defined by $p(x)=x^4+x+1$ over $\mathbb{Z}_2$.

Example 9.4 Let's take $n=15$, $p=2$. Since $n=p^4-1$, the nth roots of unity are exactly $E^{\times}$, where $E=\mathbb{F}_{16}$, which means we just need to construct E and go from there. In particular, we will take γ to be a primitive element of E over $\mathbb{Z}_p$.

To construct E, choose an irreducible polynomial of degree 4. Let's take $p(x)=x^4+x+1$. Then we construct $\mathbb{F}_{16}=F[x]/\langle p(x)\rangle$ and compute the multiplicative and additive forms of all elements in Table 9.1.

Now let's see what cyclic codes of length $n=15$ we can construct with this information.

First approach: trial and error. Say $g(x)=p(x)=x^4+x+1$; this gives a $(15,11)$ cyclic code. The roots of g, by the table, are $\alpha,\alpha^2,\alpha^4,\alpha^8$. So taking $\gamma=\alpha$, we count exactly 2 consecutive roots[1], so $d_{min}>2$. Since the generator already has weight 3, we conclude that $d_{min}=3$.

Second approach: by design. Say we wish to construct a code with designed distance 7. So we'd need a polynomial with 6 consecutive roots, say $\{\alpha,\alpha^2,\cdots,\alpha^6\}$.

To do this, take the set of minimal polynomials of α, α^2, $\cdots$, α^6. They are $p(x)$, $r(x)$ and $t(x)$.

So if we were to choose $g(x)=p(x)r(x)t(x)$, we would have a polynomial with 6 consecutive roots, and so for the code $C=\langle g(x)\rangle$, we have $d_{min}\geq 7$.

How big is it? Since $\deg(p(x))=4$, $\deg(r(x))=4$ and $\deg(t(x))=2$, we have that $\deg(g(x))=10$ and so this code is a binary 3-error-correcting $(15,5)$ cyclic code. (We can verify directly, by multiplying

[1]Of course γ is not the only primitive 15th root of unity. Could there be a different choice for which we'd get a different count of the number of consecutive roots? Here, it is easy to argue that this does not occur; if we had 3 consecutive roots then two of the pairwise quotients of the roots would be equal; but they are all distinct.

out the generator, that again this code contains an element of weight exacty 7.) ■

R From Table 9.1, we can find all the primitive kth roots of unity, for all divisors of 15, that is, for all $k \in \{3,5,15\}$. Thus we can work out a lower bound on d_{min} for any binary cyclic code of length 3, 5 or 15 using this table.

This second approach is what defines BCH codes.

Definition 9.5 Let F be a finite field of characteristic p and let n,d,ℓ be integers such that $2 \leq d \leq n$, $\gcd(n,p) = 1$ and $\ell \geq 0$. Let γ denote a primitive nth root of unity. A *BCH code* of *designed distance* d is a cyclic code of length n over F with generator

$$g(x) = lcm(m_\ell(x), m_{\ell+1}(x), \cdots, m_{\ell+d-2}(x))$$

where $m_k(x)$ denotes the minimal polynomial of γ^k.

This approach provides a streamlined way to produce codes: create a table like Table 9.1, and look for a consecutive sequence of powers of γ whose minimal polynomials have a least common multiple of low degree (hence giving a higher-dimensional cyclic code).

9.4 A new kind of parity check matrix for BCH codes

We deduce a new way of detecting codewords.

Lemma 9.6 Let C be a BCH code, with notation as in Definition 9.5. Then $f(x) \in C$ if and only if $f(\gamma^i) = 0$ for each $i = \ell, \ell+1, \cdots, \ell+d-2$.

Proof. During the proof of Theorem 9.2 we established that each of the γ^i, being roots of $g(x)$, are roots of $c(x)$ for all $c(x) \in C$.

Conversely, if $f(\gamma^k) = 0$, then since $m_k(x)$ is the minimal polynomial of γ^k, we have that $m_k(x)$ divides $f(x)$. Thus the hypothesis implies that the least common multiple of all these minimal polynomials divides $f(x)$, which is saying that f is a multiple of g. Thus $f(x) \in C$. ■

An immediate consequence is that the following is another "parity check matrix" for C. Define

$$H = \begin{bmatrix} 1 & \gamma^\ell & \gamma^{2\ell} & \cdots & \gamma^{(n-1)\ell} \\ 1 & \gamma^{\ell+1} & \gamma^{2(\ell+1)} & \cdots & \gamma^{(n-1)(\ell+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma^{\ell+d-2} & \gamma^{2(\ell+d-2)} & \cdots & \gamma^{(n-1)(\ell+d-2)} \end{bmatrix}. \tag{9.1}$$

Then for any vector $f \in F^n$, identified with a polynomial $f(x) \in F[x]/\langle x^n - 1\rangle$, we have

$$Hf = (f(\gamma^\ell), \cdots, f(\gamma^{\ell+d-2})).$$

Thus $Hf = 0$ if and only if $f \in C$. However, this is not at all like our "usual" parity check matrices:

For one, the entries of this matrix are in the field $\mathbb{F}_q$ that contains γ; in general this is not the base field $\mathbb{Z}_p$. To interpret H as a matrix over $\mathbb{Z}_p$, choose a linear isomorphism $\mathbb{F}_q \cong \mathbb{Z}_p^m$, and replace each entry in H with the corresponding column vector. So although H has size $(d-1) \times n$ as a matrix with entries in $\mathbb{F}_q$, it "really" has size $(d-1)m \times n$ as a matrix with entries in $\mathbb{Z}_p$.

For another: we previously defined our parity check matrices to have size $(n-k) \times n$, for an (n,k) code; but the matrix H above is unlikely to be of the correct size (no matter how we write it). What's happening is that $(d-1)m \geq n-k$, and therefore in fact some of these rows are linearly dependent. Note that this does not affect the ability of H to detect errors; it just implies our H is bigger than it needs to be.

We will nevertheless call H a parity check matrix for the BCH code C.

9.5 Reed-Solomon codes

Definition 9.7 Let F be a field with q elements. A *Reed-Solomon code* is a BCH code of length $n = q-1$ over F of minimal distance at least d whose generator has the form

$$g(x) = \prod_{i=\ell}^{\ell+d-2} (x-\gamma^i)$$

for some ℓ, where γ is a primitive element of F.

This is just a very special case of BCH codes, where the primitive nth root of unity already live in your base field, and so their minimal polynomials are just linear.

Consider the following example.

■ **Example 9.8** Let $p = 7$. A primitive root of $\mathbb{F}_7$ is 3, since $\mathbb{F}_7 = \mathbb{Z}_7$ and the powers of 3 give all nonzero elements of $\mathbb{Z}_7$. Thus $\gamma = 3$ has order exactly 6, which makes it a primitive 6th root of unity, and so $x^6 - 1 = \prod_{i=1}^{6}(x-i)$. Note that $3^2 = 9 = 2$. So

$$g(x) = (x-3)(x-2)$$

is a generator for a $(6,4)$ cyclic code. Since its roots are two consecutive powers of a primitive 6th root of unity, we deduce from the theorem that $d_{min} > 2$. Since $g(x) = x^2 + 2x + 6$ has weight 3, we deduce that $d_{min} = 3$, and this is a single-error-correcting code. ■

Proposition 9.9 Reed-Solomon codes are MDS, that is, they satisfy the Singleton bound.

(The proof is left as an exercise.)

Where Reed-Solomon codes are handy is the following kind of example.

■ **Example 9.10** Let's take $F = \mathbb{F}_{2^8}$. So we can choose a linear basis for F over $\mathbb{Z}_2$ and thereby write each element of F as a vector in $\mathbb{Z}_2^8$ (an isomorphism which preserves addition but has no idea about multiplication in $\mathbb{Z}_2^8$). So we can think of each symbol in a codeword as being a byte, rather than a bit.

Now F has $q = 2^8 = 256$ elements, so we can create a Reed-Solomon $(255,k)$ code C for any choice of k, such that $d_{min} = 255 - k + 1$. A popular choice is $(255,223)$, which has $d_{min} = 33$ so corrects 16 errors.

Each codeword in C is 255 bytes long (or 2048 bits). What the code corrects is 16 *byte errors*, not 16 *bit errors*. That is, it can correct errors or the form

- 1 bit error in 16 separate bytes, or
- 16 consecutive bytes completely trashed ("burst error").

■

9.6 Decoding BCH codes: the theory

We know how to decode any linear code via syndrome decoding, but at the same time we realize that our algorithm is most efficient in the case of single-error-correcting codes. Let's outline a decoding algorithm for BCH codes which, for sufficiently large n (say $n > 25$) is more efficient than any syndrome table-lookup.

Setup: Suppose C is an (n,k) BCH code of designed distance d; then it is t-error-correcting where $t = \lfloor (d-1)/2 \rfloor$. Suppose that γ is a primitive nth root of unity and to make the indexing simpler, let's say that the largest string of consecutive roots of the generator polynomial $g(x)$ is γ, γ^2, $\cdots$, γ^{d-1}, with $d \geq 2t+1$.

The approach of this decoding algorithm: Suppose we receive a vector $v \in \mathbb{Z}_p^n$, viewed as a polynomial $v(x) \in \mathbb{Z}_p[x]/\langle x^n - 1 \rangle$. Then there is some $c \in C$ and an *error polynomial* e such that

$$v(x) = c(x) + e(x).$$

If v is the result of $r \leq t$ errors, then we can write the error polynomial as

$$e(x) = e_{j_1} x^{j_1} + \cdots + e_{j_r} x^{j_r} \tag{9.2}$$

for r distinct exponents j_i and unknown coefficients e_{j_i}. (If $p = 2$, then all these $e_{j_i} = 1$, and things are a bit simpler!) This polynomial is not known; we need to find a way to solve for

- r,
- the r exponents, and
- the r nonzero coefficients.

What we have: Using H as in (9.1) (with $\ell = 1$), and given that $c \in C$ so that $c(\gamma^i) = 0$ for all $1 \leq i \leq d-1$, the syndrome S_i is

$$S_i = v(\gamma^i) = c(\gamma^i) + e(\gamma^i) = e(\gamma^i),$$

which we can calculate from the received vector v.

We have $d-1$ syndromes, and by definition of error correction, $d-1 \geq 2t$. So we have at least $2t$ equations, and $2r \leq 2t$ unknowns; this looks good.

The problem: nonlinearity. Our equations are nonlinear (in the variables for which we want to solve)! A key rule of thumb is that all we can ever solve painlessly are linear equations.

To see they are nonlinear, it's easier to write z_i for γ^{j_i}, and y_i for e_{j_i}, so that we have

$$\begin{aligned} S_1 &= y_1 z_1 + \cdots + y_r z_r \\ S_2 &= y_1 z_1^2 + \cdots + y_r z_r^2 \\ \vdots \quad & \qquad \vdots \\ S_{2t} &= y_1 z_1^{2t} + \cdots + y_r z_r^{2t}. \end{aligned} \tag{9.3}$$

Note, however, that for any r consecutive equations we have a square linear system $\vec{S} = V_r \vec{y}$, where $\vec{S} = (S_m, \cdots, S_{m+r})$, $\vec{y} = (y_1, \cdots, y_r)$ and $V_{r,m}$ is the matrix

$$V_{r,m} = \begin{bmatrix} z_1^m & \cdots & z_r^m \\ z_1^{m+1} & \cdots & z_r^{m+1} \\ \vdots & & \vdots \\ z_1^{m+r} & \cdots & z_r^{m+r} \end{bmatrix} = V_r(z_1, \cdots, z_r) D(z_1^m, \cdots, z_r^m) \tag{9.4}$$

where $V_r(z_1, \cdots, z_r)$ is a Vandermonde matrix and $D(z_1^m, \cdots, z_r^m)$ is a diagonal matrix with z_i^m's along the diagonal. By Proposition 9.1 we conclude that if the z_i are distinct (and none are 0), this matrix has nonzero determinant and so the system will have a unique solution for the y_i.

But how do we find the z_i?

The trick: the error locator polynomial The special (and surprising) trick is to define a new polynomial, called the *error locator polynomial* $s(x)$. (We don't know what it is — we'll have to solve for its coefficients and guess its degree.) We define it by

$$s(x) = (1 - xz_1)(1 - xz_2) \cdots (1 - xz_r) \tag{9.5}$$

so that it is a polynomial whose roots are exactly the inverses of the r distinct $z_i = \gamma^{j_i}$ that we're trying to find.

This polynomial has constant term 1, but we do not know the rest of the coefficients. Write

$$s(x) = 1 + s_1 x + \cdots + s_r x^r$$

for r unknowns $s_1, \cdots, s_r$. We claim we can find these coefficients s_j, as follows.

Observe that we have, for each $i \in \{1, 2, \cdots, r\}$ and $j \in \{1, 2, \cdots, 2t - r\}$,

$$\begin{aligned} 0 &= s(z_i^{-1}) \\ &= y_i z_i^{j+r} s(z_i^{-1}) \quad \text{multiplying by a nonzero element} \\ &= y_i z_i^{j+r}(1 + s_1 z_i^{-1} + \cdots + s_r z_i^{-r}) \\ &= y_i z_i^{j+r} + s_1 y_i z_i^{j+r-1} + \cdots + s_r y_i z_i^{j}. \end{aligned}$$

Now, we sum over i to get

$$\begin{aligned}
0 &= \sum_{i=1}^{r}(y_i z_i^{j+r} + s_1 y_i z_i^{j+r-1} + \cdots + s_r y_i z_i^{j}) \\
&= \sum_{i=1}^{r} y_i z_i^{j+r} + s_1 \sum_{i=1}^{r} y_i z_i^{j+r-1} + \cdots + s_r \sum_{i=1}^{r} y_i z_i^{j} \\
&= S_{j+r} + s_1 S_{j+r-1} + \cdots + s_r S_j.
\end{aligned}$$

Notice that since $j + r \leq 2t$, these are all valid syndromes, for every $j \in \{1, 2, \cdots, 2t - r\}$: each of the S_k are known values we have previously computed.

This gives us an inhomogeneous system of linear equations for the variables $s_1, \cdots, s_r$, which we can write in matrix form as

$$S\,\vec{s} = \begin{bmatrix} S_1 & S_2 & \cdots & S_r \\ S_2 & S_3 & \cdots & S_{r+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_r & S_{r+1} & \cdots & S_{2r-1} \end{bmatrix} \begin{bmatrix} s_r \\ s_{r-1} \\ \vdots \\ s_1 \end{bmatrix} = \begin{bmatrix} -S_{r+1} \\ -S_{r+2} \\ \vdots \\ -S_{2r} \end{bmatrix}. \tag{9.6}$$

Proposition 9.11 Let D be a diagonal matrix with the values $y_i z_i$ along the diagonal. Then the matrix S of (9.6) is given by

$$S = VDV^T,$$

where $V = V_r(z_1, \cdots, z_r)$ is a Vandermonde matrix of size r.

Proof. From (9.3) and (9.4), we have that

$$\begin{bmatrix} S_m \\ \vdots \\ S_{r+m} \end{bmatrix} = V \begin{bmatrix} y_1 z_1^m \\ \vdots \\ y_r z_r^m \end{bmatrix}.$$

These are the various columns of the matrix S, as m goes from 1 to r. Factoring out the common factor $y_i z_i$ from each row produces V^T; thus we have

$$S = V \begin{bmatrix} y_1 z_1 & \cdots & y_1 z_1^r \\ \vdots & & \\ y_r z_r & \cdots & y_r z_r^r \end{bmatrix} = VDV^T,$$

as required. ■

It follows that if the z_i are distinct, and the y_i nonzero (i.e. if we guessed r correctly) then this system has a unique solution. Conversely, it means that we should choose r to be the largest value $\leq t$ for which S is an invertible matrix.

We can then solve the system (9.6), using any favourite method, to get the coefficients s_i, and consequently, write down the error locator polynomial $s(x)$.

What does knowing the error locator polynomial give us? Given $s(x)$, we first want to find its roots. When the order of γ is not too large, we can find the roots by just plugging in powers of γ until we find r different γ^i such that $s(\gamma^i) = 0$. More generally, there exist classic multi-purpose efficient implementations of root-finding algorithms (for example, [Ber68, Ber70, CZ81]) and improving these for various applications is an active area of research.

For each root you find, in the form γ^{n-k}, remember that its inverse γ^k is one of the $z_i = \gamma^{j_i}$. Therefore for each k such that γ^{n-k} is a root of $s(x)$, k is one of the error positions j_i. Therefore finding all the roots of $s(x)$ gives all the error positions.

If our field is not binary, then we need to find the $y_i = e_{j_i}$, which are the coefficients of the error polynomial $e(x)$ in (9.2). To do so, note that (9.3) is now just a linear system in just the unknowns y_i, and so these values can again be found by linear algebra.

R We have given a decoding algorithm which involves solving a pair of linear systems of size about t. The complexity of row reduction to solve such a system is in the order of t^3. For large $t < n$ this is still vastly better than a table lookup of the p^{n-k} syndromes; but in fact even more efficient algorithms have been developed. In particular, a famous one by Berlekamp and Massey, can be implemented with shift registers (which we have already seen are very efficient).

We can summarize this as an algorithm (where we now allow $g(x)$ to have any $d-1$ consecutive roots).

Algorithm 9.12 — BCH decoding. Given a BCH code $C = \langle g(x) \rangle$ of length n over a field F, where $g(x) = \operatorname{lcm}\{m_a(x) \mid a \in \{\gamma^{\ell+1}, \gamma^{\ell+2}, \cdots, \gamma^{\ell+d-1}\}\}$ for some generator γ of an extension field E of F that contains μ_n, and given a received vector $v(x) \in F[x]/\langle x^n - 1\rangle$, we decode $v(x)$ as follows:

1. Calculate the syndromes $S_1, \cdots, S_{d-1}$ by evaluating

$$S_i = v(\gamma^{\ell+i}) \in E.$$

2. Choose the largest value of $r \le t$ for which the matrix

$$S = \begin{bmatrix} S_1 & S_2 & \cdots & S_r \\ S_2 & S_3 & \cdots & S_{r+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_r & S_{r+1} & \cdots & S_{2r-1} \end{bmatrix}$$

   is invertible.
3. Solve for the unknowns $s_1, \cdots, s_r$ in the equation

$$S \begin{bmatrix} s_r \\ s_{r-1} \\ \vdots \\ s_1 \end{bmatrix} = \begin{bmatrix} -S_{r+1} \\ -S_{r+2} \\ \vdots \\ -S_{2r} \end{bmatrix}.$$

4. Find the roots of $s(x) = 1 + s_1 x^1 + \cdots + s_r x^r$ in E.
5. Find the powers j_i such that the roots of $s(x)$ are $\gamma^{-j_1}, \gamma^{-j_2}, \cdots, \gamma^{-j_r}$.

6. Set $z_i = \gamma^{j_i}$ for each i. Now solve

$$\begin{bmatrix} z_1^{\ell+1} & \cdots & z_r^{\ell+1} \\ z_1^{\ell+2} & \cdots & z_r^{\ell+2} \\ \vdots & & \vdots \\ z_1^{\ell+r} & \cdots & z_r^{\ell+r} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \cdots \\ S_r \end{bmatrix}$$

   for the unknowns $y_1, \cdots, y_r$.
7. Set $e(x) = y_1 x^{j_1} + \cdots + y_r x^{j_r}$.
8. Compute $c(x) = v(x) - e(x)$.
9. Check that $c(x) \in C$ by evaluating it on the roots $\gamma^{\ell+i}$ for $1 \leq i < d$. If $c(x) \notin C$, then $v(x)$ was the result of more than t errors.

9.7 Decoding BCH codes: examples

Performing this algorithm by hand is quite torturous! Nevertheless, let's look at a couple of examples, and explore a few calculation tricks to bear in mind.

■ **Example 9.13** For example, let $g(x) = 1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10}$ be a generator polynomial for a $(15, 5)$ binary BCH code.

We begin by factoring $g(x) = r(x)s(x)t(x)$, with notation as in Table 9.1, so the consecutive roots are α^9, $\cdots$, α^{14}, and we deduce that the designed distance is $d = 6 + 1$; since the weight of $g(x)$ is 7 we conclude that $d_{min} = 7$. Thus this code can correct up to 3 errors.

Suppose the word $v = 101101011001001$ is received, and let's determine the correct word that was sent, by solving for the error locator polynomial and finding its roots. Write

$$v(x) = 1 + x^2 + x^3 + x^5 + x^7 + x^8 + x^{11} + x^{14}.$$

Let's copy the table describing $\mathbb{F}_{16}$ here for ease of reference.

mult. form	add. form	mult. form	add. form
-	0	α^7	$1+\alpha+\alpha^3$
1	1	α^8	$1+\alpha^2$
α	α	α^9	$\alpha+\alpha^3$
α^2	α^2	α^{10}	$1+\alpha+\alpha^2$
α^3	α^3	α^{11}	$\alpha+\alpha^2+\alpha^3$
α^4	$1+\alpha$	α^{12}	$1+\alpha+\alpha^2+\alpha^3$
α^5	$\alpha+\alpha^2$	α^{13}	$1+\alpha^2+\alpha^3$
α^6	$\alpha^2+\alpha^3$	α^{14}	$1+\alpha^3$

Table 9.2: Addition and multiplication in the field $\mathbb{F}_{16}$, defined by $p(x) = x^4 + x + 1$ over $\mathbb{Z}_2$.

1. The first step is to calculate all the syndromes, which we can do using the 6×15 parity check matrix

$$H = \begin{bmatrix} 1 & \alpha^9 & \alpha^{18} & \cdots \\ 1 & \alpha^{10} & \alpha^{20} & \cdots \\ \vdots & \vdots & \vdots & \\ 1 & \alpha^{14} & \alpha^{28} & \cdots \end{bmatrix} = \begin{bmatrix} 1 & \alpha^9 & \alpha^{3} & \cdots \\ 1 & \alpha^{10} & \alpha^{5} & \cdots \\ \vdots & \vdots & \vdots & \\ 1 & \alpha^{14} & \alpha^{13} & \cdots \end{bmatrix}.$$

Thus for example,

$$\begin{aligned} S_1 &= v(\alpha^9) \\ &= 1+\alpha^3+\alpha^{12}+1+\alpha^3+\alpha^{12}+\alpha^9+\alpha^6 \\ &= \alpha^6+\alpha^9 \\ &= \alpha+\alpha^2 \\ &= \alpha^5 \end{aligned}$$

where we have used that $\alpha^{15} = 1$, that $2 = 0$ in $\mathbb{F}_{16}$, and Table 9.2 to simplify the answer in the last lines.

In practice, it is much easier to use the fact that if

$$v(x) = q(x)m(x) + t(x)$$

where $m(x)$ is the minimal polynomial of γ, then $v(\gamma) = t(\gamma)$ — because the degree of t is much lower than that of v, in practice.

The only minimal polynomials we need to worry about here are those occurring in the factorization of $g(x)$, and by performing long division we find that

$$\begin{aligned} v(x) &= q_1(x)r(x) + x^3 + x^2 + 1 \\ v(x) &= q_2(x)s(x) + x^3 + x^2 + x \\ v(x) &= q_3(x)t(x) + 1. \end{aligned}$$

So we further compute, with much less hassle, that

$$\begin{aligned} S_2 &= v(\alpha^{10}) = 1 \\ S_3 &= v(\alpha^{11}) = \alpha^{13} \\ S_4 &= v(\alpha^{12}) = \alpha^{10} \\ S_5 &= v(\alpha^{13}) = 1+\alpha^3 = \alpha^{14} \\ S_6 &= v(\alpha^{14}) = 1+\alpha+\alpha^3 = \alpha^7. \end{aligned}$$

2. So we construct the matrix for the maximum number of errors, which is:

$$\begin{bmatrix} \alpha^5 & 1 & \alpha^{13} \\ 1 & \alpha^{13} & \alpha^{10} \\ \alpha^{13} & \alpha^{10} & \alpha^{14} \end{bmatrix}$$

but this has determinant zero, meaning the system is overdetermined, which means we chose too high a degree for our polynomial. So there are not 3 errors, but perhaps 2.

3. Thus we take $r = 2$ and we want to solve (9.6), which is:

$$\begin{bmatrix} \alpha^5 & 1 \\ 1 & \alpha^{13} \end{bmatrix} \begin{bmatrix} s_2 \\ s_1 \end{bmatrix} = \begin{bmatrix} \alpha^{13} \\ \alpha^{10} \end{bmatrix}.$$

Thus

$$\begin{bmatrix} s_2 \\ s_1 \end{bmatrix} = \begin{bmatrix} \alpha^5 & 1 \\ 1 & \alpha^{13} \end{bmatrix}^{-1} \begin{bmatrix} \alpha^{13} \\ \alpha^{10} \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha^7 \end{bmatrix}$$

where we have used the formula for the inverse and spent some time simplifying.

4. Thus $s_1 = \alpha^7$ and $s_2 = 1$ so our error locator polynomial is

$$1 + \alpha^7 x + x^2.$$

We find that α is a root; dividing by $x - \alpha$ yields $x - \alpha^{14}$ so the two roots are α and α^{14}.

5. So $z_1^{-1} = \alpha^{14}$ and $z_2^{-1} = \alpha$, which means $z_1 = \alpha$ and $z_2 = \alpha^{14}$. (Note that it was just luck that they were inverses of one another.) Now $z_i = \alpha^{j_i}$ so we have $\{j_1, j_2\} = \{1, 14\}$.

6. Since our field is binary, we're done (all the coefficients y_i must be 1). We could check by multiplying

$$\begin{bmatrix} z_1^9 & z_2^9 \\ z_1^{10} & z_2^{10} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha^9 + \alpha^6 \\ \alpha^{10} + \alpha^5 \end{bmatrix} = \begin{bmatrix} \alpha + \alpha^2 \\ 1 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix},$$

as required.

7. This says that our error polynomial is

$$e(x) = x + x^{14}.$$

8. The correct codeword is

$$c(x) = v(x) - e(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 + x^{11}.$$

9. We verify that $r(x)$, $s(x)$ and $t(x)$ each divide evenly into $c(x)$ with no remainder, which shows that c vanishes on all the roots of r, s, t; in particular, c vanishes on $\alpha^9 \cdots \alpha^{14}$ and therefore lies in C. ■

■ **Example 9.14** Let's try a more straightforward example that is not binary, using a Reed-Solomon $(10, 6)$ code over $\mathbb{Z}_{11}$.

We have $\mathbb{Z}_{11}^{\times} = \{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$, where we have listed these elements as successive powers of the generator 2 for convenience in what follows. Suppose we choose the Reed-Solomon code generated by

$$g(x) = (x-2)(x-4)(x-8)(x-5);$$

since these correspond to the minimal polynomials of 4 consecutive roots, our designed distance is 5 and we can correct (at least) two errors.

Suppose our received codeword is 0480185310 which corresponds to

$$v(x) = 4x + 8x^2 + x^4 + 8x^5 + 5x^6 + 3x^7 + x^8.$$

1. We evaluate (using a spreadsheet, for example) that

$$\begin{aligned}
S_1 &= v(2) = 7\\
S_2 &= v(4) = 1\\
S_3 &= v(8) = 5\\
S_4 &= v(5) = 0
\end{aligned}$$

so our received vector is not a codeword.

2. Here, $t = 2$ and indeed the 2×2 matrix is invertible, so $r = 2$.

3. We solve

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} s_2 \\ s_1 \end{bmatrix} = \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix}$$

using the matrix inverse

$$\begin{bmatrix} s_2 \\ s_1 \end{bmatrix} = \begin{bmatrix} 7 & 1 \\ 1 & 5 \end{bmatrix}^{-1} \begin{bmatrix} -5 \\ 0 \end{bmatrix} = \frac{1}{1} \begin{bmatrix} 5 & -1 \\ -1 & 7 \end{bmatrix} \begin{bmatrix} -5 \\ 0 \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \end{bmatrix}.$$

4. Thus $s(x) = 1 + 5x + 8x^2$. We can use the quadratic formula to find its roots:

$$z_i^{-1} = \frac{-5 \pm \sqrt{25 - 32}}{16} = \frac{6 \pm \sqrt{4}}{5} = 9(6 \pm 2) = 3, 6.$$

5. We have $6 = 2^9 = 2^{-1}$ and $3 = 2^8 = 2^{-2}$ so $j_1 = 1$ and $j_2 = 2$.

6. Thus $z_1 = 2$ and $z_2 = 4$. Since here $\ell = 0$ we solve

$$\begin{bmatrix} z_1 & z_2 \\ z_1^2 & z_2^2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

by using matrix inversion to get

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 4 & 16 \end{bmatrix}^{-1} \begin{bmatrix} 7 \\ 1 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 5 & -4 \\ -4 & 2 \end{bmatrix} \begin{bmatrix} 7 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \end{bmatrix}.$$

7. Thus $e(x) = 4x^1 + 8x^2$.

8. This yields $c(x) = v(x) - e(x) = x^4 + 8x^5 + 5x^6 + 3x^7 + x^8$, which is actually $x^4 g(x)$ so clearly in C.

This was a lot of work — but far less work than creating and searching through a table of syndromes as in Section 5.3 (which would have $|F|^{n-k} = 11^4 = 14{,}641$ rows).

■

9.8 Further topics of interest

BCH codes were classically used in several important networks, including:

- European information communication systems, with a $(255, 231)$ BCH code which detects 6 errors;
- INTELSAT-V satellite communications, with a $(128, 112)$ code;
- cell phone circa 1997 used a BCH $(48, 36)$ code with $d_{min} = 5$.

Their value is in their abundance and ease of construction; for any given block size and error-correction desired, within reason, you can construct such a code. Their algebraic structure also makes them ideal for applications requiring families of codes that are scalable and whose characteristics and decoding properties are provable (eg: cryptography).

One can purchase implementations of BCH codes, or find some on-line [MZ23]. They are inefficient for small n, however, because there's quite a bit of overhead to include the polynomial arithmetic and field extensions.

While BCH codes have excellent error rates, they do not approach the Shannon channel capacity bound (Section 3.2). The two most common codes in use that do approach this capacity bound are the LDPC (low-density parity check) linear codes based on random graphs (discovered in 1960 but they were impractical before modern computing power) and turbo codes (discovered in 1993), which use probabilistic tools (like Bayesian inference) to decode, have excellent performance, and are very widely used in applications.

Some related topics of interest, for those who want to delve deeper into codes, include:

- decoding algorithms, including particularly those due to Berlekamp, which exploit features of polynomial arithmetic to be extremely efficient;
- codes for different uses than presented here: for data compression or cryptography;
- other algebraic codes (quadratic residue codes, Goppa codes, algebraic geometry codes);
- space-time codes (for multiple antenna / multiple receiver systems).

9.9 Exercises

1. Write down the elementary matrix A such that the elementary row operation "add c times row (i) to row (j)" is given by left multiplication by A, that is: such that the result of doing this elementary row operation on B produces the matrix AB. Show that $\det(A) = 1$.
2. Write down the elementary matrix C such that the elementary row operation "multiply row (i) by $c \in F^{\times}$" is given by left multiplication by C, that is: such that the result of doing this elementary row operation on B produces the matrix CB. Show that $\det(A) = c$.
3. Write down the elementary matrix P such that the elementary row operation "swap rows (i) and (j)" is given by left multiplication by P, that is: such that the result of doing this elementary row operation on B produces the matrix PB. Show that $\det(P) = -1$.
4. Show that *right* multiplication of a matrix B by the matrices A, C and P of the previous exercises performs the corresponding *column operations* on B.
5. Prove that if $\gcd(n, p) = 1$ then there exists $m \geq 1$ such that $n | (p^m - 1)$. *Hint: Consider the powers of p mod n.*
6. Suppose F is a field of characteristic p. Prove that if p divides n, then $x^n - 1$ has repeated roots, and in fact you cannot find any factors of $x^n - 1$ which are not already factors of a $x^k - 1$ for some $k < n$.
7. Prove that the matrix H in (9.1) is a parity check matrix for C.

8. Prove that over $\mathbb{Z}_2$, if $\alpha \in \mathbb{F}_{2^m}$ is a root of an irreducible polynomial $p(x)$, then so are α^2, α^4, $\alpha^8, \cdots$.
9. Prove that for any positive integers m and $e \leq 2^{m-1}-1$, there is a binary BCH code of length $n = 2^m - 1$ that is e-error-correcting and has dimension $k \geq n - me$ (that is, $n-k \leq me$). *Hint: Working over $\mathbb{F}_{2^m}$, use the preceding exercise to show that the maximum number of distinct minimal polynomials you'll need to construct such a code is e; show also that the degrees of these minimal polynomials are bounded by m.*
10. Prove that Reed-Solomon codes are MDS.
11. For the code in Example 9.14, decode

$$v(x) = 1 + 9x + 4x^2 + 7x^3 + x^4 + 7x^5 + 2x^6.$$

12. For the code in Example 9.14, decode

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

III

Cryptography

10. Public-Key Cryptography

We now turn to another aspect of communications — security. Once again, algebraic structures underlie some of the most important algorithms in use today.

Suppose that Alice wants to send a secret message to Bob, but knows that Eve the eavesdropper may intercept it. How can she encode the message so that Eve does not gain information about its content but Bob can fully recover the message?

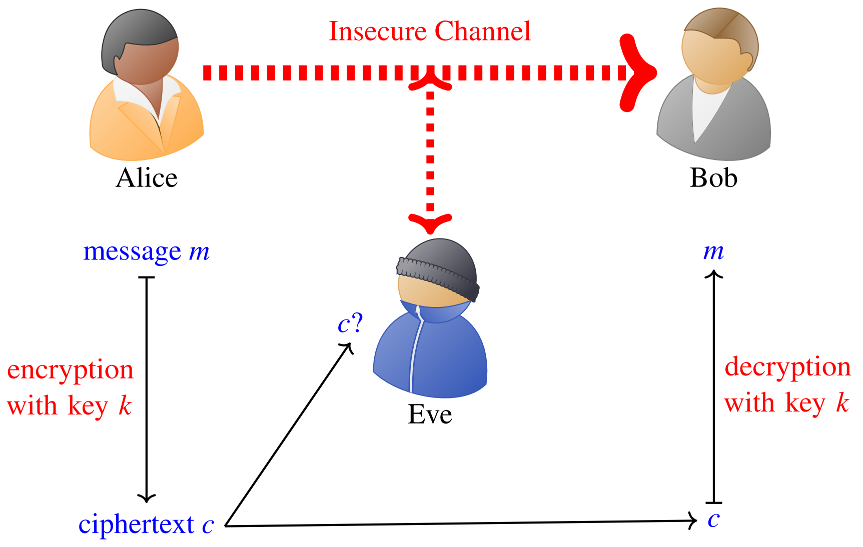


Figure 10.1: Alice uses a secret shared key k to encrypt her message m as the ciphertext c. She transmits it to Bob over an insecure channel, where it is intercepted by Eve. Bob, using k, is able to decrypt c to recover m, but Eve cannot.

10.1 Perfect secrecy: the one-time pad

Claude Shannon proved, in a paper [Sha49] extending his theory of information to the question of cryptography, that there is effectively only one way to achieve perfect secrecy : the *one-time pad*.

Fix a message length n. Alice and Bob agree in advance upon a secret key k that is chosen uniformly at random from $\mathbb{Z}_2^n$. To encrypt a plaintext message $m \in \mathbb{Z}_2^n$, Alice computes the ciphertext

$$c = m + k \in \mathbb{Z}_2^n.$$

She sends the vector c to Bob, who can decrypt it by adding the vector k:

$$c + k = (m + k) + k = m + 2k = m.$$

What information does Eve gain from knowing c? By Shannon's theory of communcation: zero information. Each bit of m has a 50% probability of having been changed, so knowing the output bit does not reveal anything about the input bit.[1]

R This doesn't mean Eve is necessarily thwarted: perhaps the timing of the communication tells Eve all she needed to know, for example. The encryption algorithms we discuss will always just be the first (and hopefully by far the most secure) layer of information privacy.

There are a couple of downsides to the one-time pad:

(i) The secret key needs to be at least as long as the message; and
(ii) Alice and Bob need to secretly share k using some other means of communication.

Since perfect secrecy is possible only with a one-time pad, the next-best thing is to consider secrecy under the realistic constraint of *limited computing power*. That is, instead of asking for it to be *impossible* for Eve to determine the message, we ask that the probability that she can learn ε about m (in the Shannon sense of information) is negligible (for example, a 1 in 2^{40} chance with an investment of 2^{40} of computing time and space).

R To convey this accurately requires the tools of probability and complexity theory — neither of which we assume in this course! Therefore, in this chapter, we restrict ourselves to describing these notions at an intuitive level, using phrases like "efficient algorithm," "computationally infeasible" and "hard mathematical problem" — leaving our focus on the algebraic tools involved.

So to resolve (i), we can use a smaller key k as input to an efficient symmetric cipher, like AES (Advanced Encryption Standard). This is an algorithm that takes the key and a message of arbitrary length as input, then expands the key to whatever length it needs and scrambles the message block by block, using the key. It is computationally infeasible to recover the message without the key.[2]

[1] If Alice re-uses k to encrypt another message, then this no longer holds; hence the name "one-time pad".

[2] As of now, that is. The previous standard was called Data Encryption Standard (DES), developed in 1977, and which had not sufficiently anticipated the acceleration of computing power of the following decades. While implementers used "triple DES" as a patch, NIST (National Institute of Standards and Technology, USA) ran a call for a new standard at the end of the century, choosing AES in 2001.

But (ii) seems like a major obstacle: if all lines of communication between Alice and Bob are insecure, surely they cannot exchange *any* secret information, let alone a secret key? Wasn't this the whole point? This is the setting for our mathematical discussion of cryptography.

10.2 New directions in cryptography

In 1976, Whit Diffie and Martin Hellman published a paper entitled "New directions in cryptography" [DH76]. Like Shannon's paper of 30 years earlier, it revolutionized the mathematical theory of communication, this time laying the foundation for an entirely new approach to ensuring its privacy and security: public key cryptography.[3] Unlike Shannon, however, their initial paper did not contain a solution to the central problem, but rather a challenge to the community to find one.

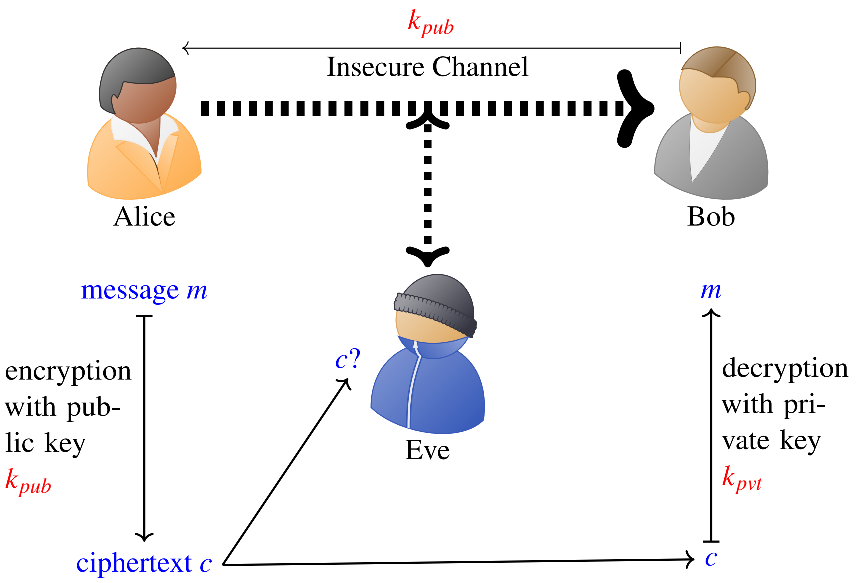

k_{pub}
Insecure Channel
Alice
Bob
message m
encryption with public key k_{pub}
ciphertext c
c?
Eve
m
decryption with private key k_{pvt}
c

Figure 10.2: Bob generates a key pair (k_{pub}, k_{pvt}) such that anyone with k_{pub} can encrypt a message to him but only he, with access to k_{pvt}, can decrypt it.

The concept of public-key cryptography is elegant, and modelled on the idea of a padlock: if you give people open padlocks (k_{pub}) such that only you have the key (k_{pvt}), then anyone with a padlock can securely send a message to you. This concept, now with numbers instead of padlocks, is illustrated in Figure 10.2.

Thus the requirements for a public-key cryptosystem are:

(i) Bob should be able to generate a key pair (k_{pvt}, k_{pub}), publish k_{pub} and keep k_{pvt} to himself alone;
(ii) There should be an efficient encryption algorithm E that takes the public key k_{pub} and a message m and produces as output a ciphertext c;
(iii) It should be computationally infeasible for Eve to either discover the private key or to find a

[3] In fact, Diffie and Hellman were not the first to discover this concept; James Ellis, working for the GCHQ (U.K.), was the first in 1969, but his work was classified. Similarly, other researchers in GCHQ, Malcolm Williamson and Clifford Cocks, discovered what we call the Diffie–Hellman key exchange algorithm and RSA cryptography, respectively.

preimage of the encryption function, even knowing k_{pub};

(iv) There should be an efficient decryption algorithm D that takes the private key k_{pvt} and a ciphertext c and outputs the original message m.

Part (ii) stipulates the existence of a function E; part (iii) stipulates that it is *one-way* in the sense that it is efficient to compute but extremely difficult to invert; and part (iv) stipulates that E has a *trapdoor*, that is, there exists additional information such that access to that information makes inverting it efficient.

Within two years, several such algorithms had been found, including RSA and the Diffie-Hellman key exchange. RSA relies on the difficulty of factoring a number that is the product of two very large primes, and it is not exaggerating to say that RSA is what enabled banking and commerce over the internet (where all transmitted data is subject to eavesdropping!) and thus our society today. Until around 2010, virtually every browser used RSA as the unassailable core of its cryptographic suite.[4] We won't discuss RSA further; a good reference is [Nic12, §1.5].

10.3 Diffie–Hellman key exchange and the discrete logarithm

Let G be a cyclic group of order N, generated by an element g. That is,

$$G = \langle g \rangle := \{1, g, g^2, g^3, \cdots, g^{N-1} \mid g^N = 1\}.$$

Let $x \in G$; then it is equal to g^r for some $0 \leq r < N$ and we write

$$r = \operatorname{dlog}(x).$$

The function dlog is called the *discrete logarithm* with respect to the generator g. It *does* depend on the generator, so in cases where there could be ambiguity we'll write dlog_g, specifying the choice of generator g.

■ **Example 10.1** The additive group $\mathbb{Z}_N$ is cyclic, generated by 1. In additive notation, $\langle 1 \rangle = \{0, 1, 2 \cdot 1, 3 \cdot 1, \cdots, (N-1) \cdot 1\} = \mathbb{Z}_N$. In this case $\operatorname{dlog}_1(x) = x$. ■

■ **Example 10.2** The multiplicative group $\mathbb{Z}_{13}^\times$ is cyclic, generated by 2 because

$$\mathbb{Z}_{13}^\times = \{1, 2, 2^2 = 4, 2^3 = 8, 2^4 = 3, 2^5 = 6, 2^6 = 12, 2^7 = 11, 2^8 = 9, 2^9 = 5, 2^{10} = 10, 2^{11} = 7\}$$

and $2^{12} = 1$ since $|\mathbb{Z}_{13}^\times| = 12$, by Lagrange's theorem. Thus for example $\operatorname{dlog}_2(3) = 4$, which we compute by going through the list. ■

Given a group G, and an element $g \in G$, then g generates a cyclic subgroup $\langle g \rangle \subset G$. The *discrete logarithm problem* is then: given $x \in \langle g \rangle$, find $\operatorname{dlog}_g(x)$. If the group has a sufficiently interesting structure, then this problem can be very hard — hard enough to call a one-way function!

Suppose G is a group such that the discrete logarithm problem is hard (for at least one choice of g).

The following algorithm allows Alice and Bob to calculate a secret value that Eve, limited by finite computing power, cannot obtain, even if she intercepts every aspect of their communication. This was developed by Ralph Merkle in response to the challenge posed by Diffie and Hellmann in their paper, with $G = \mathbb{Z}_p^\times$ for a sufficiently large prime p.

[4]This also made factorization a hot topic! So much progress was made in computing power and factorization algorithms that now secure RSA requires very large key sizes, and industry has shifted to more efficient Elliptic Curve Cryptography.

Algorithm 10.3 — Diffie–Hellman Key Exchange.

- Alice and Bob agree in public on a group G and element g. Let $N = |\langle g \rangle|$.
- Alice secretly chooses an integer a uniformly at random from $[2, N-1]$ and computes $A = g^a \in G$. She publishes her answer A.
- Bob secretly chooses an integer b uniformly at random from $[2, N-1]$ and computes $B = g^b \in G$. He publishes his answer B.
- Now Alice and Bob can efficiently compute their shared secret key

$$k = g^{ab}.$$

Proof of correctness. Note that Alice knows B and a so can compute $B^a = (g^b)^a = g^{ba} = g^{ab} = k$. Bob knows A and b so can compute $A^b = (g^a)^b = g^{ab} = k$. Note that this holds for any integers $a, b \in \mathbb{Z}$; therefore if the order N of g is not known, Alice and Bob can use $N = |G|$ instead. ■

The *Diffie–Hellman assumption* defining the security of this cryptographic algorithm is that Eve cannot efficiently obtain k from knowledge of g^a and g^b. So for example the Diffie–Hellman assumption does not hold for $G = (\mathbb{Z}_n, +)$ (exercise). On the other hand, it is felt to hold for $G = (\mathbb{F}_q^\times, \times)$, for q sufficiently large so as to make calculating $\text{dlog}_g(A)$ or $\text{dlog}_g(B)$ computationally infeasible. We say “felt to hold" because there is no proof that this is Eve’s ONLY option to try to get k!

R The resulting key is an element of G, but we assume there is a standard way to represent elements of G as integers or as bitstrings (so that these computations can be carried out on a computer). For example, this is relatively straightforward to do if G is a finite field (exercise).

10.4 The ElGamal public-key cryptosystem

Now suppose we go a step further. In the Diffie–Hellman key exchange, Bob created a pair of keys: b, his private key and B, his public key. He could publish this public key somewhere and everyone could use it to send him messages that only he could decrypt. This idea first appeared in a paper by Taher ElGamal [ElG85] in 1985.

Algorithm 10.4 — ElGamal public-key cryptosystem.

- Bob chooses a group G and an element g. Let $N = |\langle g \rangle|$. Bob chooses a secret integer k_{pvt} uniformly at random from $[2, N-1]$ and computes $k_{pub} = g^{k_{pvt}} \in G$. He publishes (G, g, N, k_{pub}) and keeps k_{pvt} secret.
- When Alice wants to send a message to Bob, she uses the public information (G, g, N, k_{pub}). She first converts her message to an element $m \in G$, then chooses an *ephemeral key* $a \in \mathbb{Z}$, chosen uniformly at random from $[2, N-1]$. She then computes

$$A = g^a, C = mk_{pub}^a.$$

She then destroys a. The pair (A, C), which she transmits to Bob, is the ciphertext.

- Upon receipt of (A,C), Bob recovers m as

$$m = C(A^{k_{pvt}})^{-1}.$$

Proof of correctness. First note that

$$K = A^{k_{pvt}} = g^{ak_{pvt}}$$

is the secret key from Algorithm 10.3. Therefore since $k_{pub}^{a} = g^{ak_{pvt}}$, we have

$$CK^{-1} = (mk_{pub}^{a})K^{-1} = mKK^{-1} = m \in G,$$

as required. ■

There, that was easy!

When specialized to the case that G is an *elliptic curve* over a finite field (usually $\mathbb{F}_{2^k}$ for some very large k), this is the most common public-key cryptosystem in use for browsers; it is the core of the standard protocol TLS 1.3.

10.5 Exercises

1. Prove that if Alice and Bob use a one-time pad to encrypt a message $m \in \mathbb{Z}_2^n$ then if their secret $k \in \mathbb{Z}_2^n$ is chosen uniformly an random, every output ciphertext $c \in \mathbb{Z}_2^n$ is equally likely.
2. Prove that 2 is not a generator of $\mathbb{Z}_{17}^{\times}$ but that 3 is. Given that $|\mathbb{Z}_{17}^{\times}| = 16$, what are the minimum number of calculations you need to make to prove these facts?
3. Suppose $G = \mathbb{Z}_{11}^{\times}$ and $g = 7$. Find $\mathrm{dlog}_g(10)$.
4. Suppose $G = \mathbb{Z}_{11}^{\times}$ and $g = 7$. Use Algorithm 10.4 to encrypt $m = 4$, and verify its correct decryption.

10.6 Post-quantum cryptography

Advances in number theory have brought down the complexity of integer factorization and the discrete logarithm problem to subexponential levels; correspondingly, implementations of these algorithms use larger and larger key sizes, so the systems remain secure.

The increasingly real threat is the quantum computer. In 1994, Peter Shor [Sho97] developed a quantum algorithm to factor large integers, or solve the discrete logarithm problem, in polynomial time (that is, very quickly). It depends on the existence of a sufficiently large quantum computer. While such computers were the stuff of science fiction at the time, large-scale quantum computers are quickly becoming reality, with IBM producing 433 qubit models in 2022.

Cryptosystems built on mathematical problems for which there is no known efficient algorithmic (classical or quantum) solution are called "post-quantum," or PQC, reflecting their potential for use in the quantum computing age. The testing and development of such algorithms was significantly accelerated by the public Post-Quantum Standardization exercise initiated by NIST (National Institute of Standards and Technology, USA) launched in 2017, with three rounds of submissions and the first selection of algorithms for standardization made in 2022 (and further rounds ongoing). The stakes are high: to incorporate new PQC into the infrastructure of the internet requires years of planning and coding, and there needs to be confidence that the new system is as secure as the old.

In this chapter, we present examples of PQC based on algebraic structures we have met in this course.

11. NTRU

NTRU (pronounced “en-true") was first proposed by Jeffrey Hoffstein, Jill Pipher and Joseph Silverman in 1998 [HPS98]; a good reference is [HPS14, §6.10]. It is based on the same kinds of polynomial rings as we used for cyclic codes.

11.1 NTRU algorithm

The core of the one-way trapdoor function is the following observation.

Lemma 11.1 Suppose $2 \leq p < q$ are two relatively prime integers. Then the operations “mod p" and “mod q" do not commute. More precisely: choose a set of representatives $S_p \subset \mathbb{Z}$ for the equivalence classes $\mathbb{Z}/p\mathbb{Z}$; write $\pi_p\colon \mathbb{Z}_p \to S_p \subset \mathbb{Z}$ for this map. Define S_q and π_q similarly. Then, leaving the maps $\mathbb{Z} \to \mathbb{Z}_p$ and $\mathbb{Z} \to \mathbb{Z}_q$ implicit, we have

$$\pi_p \circ \pi_q \neq \pi_q \circ \pi_p.$$

However, equality holds if $S_p \subset S_q$ and we restrict to the domain S_q.

Proof. To see that $\pi_p(\pi_q(N)) \neq \pi_q(\pi_p(N))$ in general, apply this to any integer multiple k of q. Then $\pi_q(kq) = \pi_q(0)$, so the left sides are all equal. But as $\gcd(p,q) = 1$, q is invertible mod p, so kq runs over all classes in $\mathbb{Z}/p\mathbb{Z}$. That is, $\{\pi_p(kq) \mid k \in \mathbb{Z}\} = S_p$, a contradiction.

On the other hand, suppose $S_p \subset S_q$. If $N \in S_q$, then $\pi_q(N) = N$, and similarly $\pi_p(N) \in S_p \subset S_q$ implies $\pi_q(\pi_p(N)) = \pi_p(N)$, so the equality holds in this case. ∎

■ **Example 11.2** Take $p = 2$ and $q = 101$. Let $S_p = \{0,1\}$ and $S_q = \{-50,-49,-48,\cdots,49,50\}$. Then $S_p \subset S_q$ so $\pi_q \circ \pi_p = \pi_p$, which tells us if a number is even or odd. But since q is not even, it can happen that the remainder mod q of a number N is not of the same parity as N itself. For example, if we take $N = 101 \notin S_q$, then $\pi_p(101) = 1$ whereas $\pi_p(\pi_q(N)) = \pi_p(0) = 0$. ■

Let us now present the algorithm as it appeared in the original publication; we'll spend the rest of the section analysing it. We first explain how to choose the parameters (N, p, q, d) and set some notation.

- Let $2 < p < q$ be relatively prime integers; let
$$S_p = \{0, \pm 1, \pm 2, \cdots, \pm\frac{p-1}{2}\} \quad \text{and} \quad S_q = \{0, \pm 1, \pm 2, \cdots, \pm\frac{q-1}{2}\} \tag{11.1}$$
be a fixed chosen set of representatives of the equivalence classes of $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}/q\mathbb{Z}$, respectively.
- Choose a positive integer d such that $(6d+1)p < q$, and choose a prime N such that $N > 3d$.
- Let $R = \mathbb{Z}[x]/\langle x^N - 1\rangle$; this is an integer version of the ring we used for cyclic codes. We identify R with the set of polynomials (with integer coefficients) of degree less than N, with multiplication defined by $x^N = 1$. In R, the coefficients are integers; we also define
$$R_p = \mathbb{Z}_p[x]/\langle x^N - 1\rangle, \qquad R_q = \mathbb{Z}_q[x]/\langle x^N - 1\rangle,$$
which are the kinds of rings we used for cyclic codes, if p and q are prime, for example. Note, however, that we don't require p and q to be prime (though we do require $\gcd(p,q) = 1$), so the coefficient rings of R, R_p or R_q are not necessarily fields.
- For a, b positive integers such that $a + b \le N$, let $T(a,b) \subset R$ denote the set of polynomials that have a coefficients equal to 1, b coefficients equal to -1, and the remaining coefficients 0. We can think of these as elements of R, R_p or R_q.

■ **Example 11.3** As a toy example, we may take as in [HPS14, Example 6.53]:
$$N = 7, \quad p = 3, \quad q = 41, \quad d = 2.$$
Then $|R_q| = 41^7$, $|R_p| = 3^7$ and $|T(a,b)| = \binom{7}{a}\binom{7-a}{b}$, when $a + b \le N$. ■

Algorithm 11.4 — NTRU cryptosystem: key generation. Choose (N, p, q, d) as above. Now choose two polynomials
$$f(x) \in T(d+1, d), \quad g \in T(d, d),$$
such that $f(x)$ is invertible in both R_p and R_q, that is, such that there exists $F_p(x) \in R_p$ and $F_q(x) \in R_q$ such that
$$F_p(x)f(x) = 1 \in R_p, \quad F_q(x)f(x) = 1 \in R_q.$$
Set $h(x) = F_q(x)g(x) \in R_q$; this is Bob's **public key** (together with (N, p, q, d)). Bob's **private key** is the pair $(f(x), F_p(x))$.

■ **Example 11.5** Continuing our example, we compute in this case $|T(d,d)| = \binom{7}{2}\binom{5}{2} = 210 = |T(d+1,d)|$, so we have many choices. We could choose ([HPS14, Example 7.53])
$$f(x) = x^6 - x^4 + x^3 + x^2 - 1 \in T(3,2),$$
and
$$g(x) = x^6 + x^4 - x^2 - x \in T(2,2).$$
Using the extended Euclidean algorithm, we find that the inverse of f in R_q is
$$F_q(x) = 8x^6 + 26x^5 + 31x^4 + 21x^3 + 40x^2 + 2x + 37 \in R_q,$$

and that the inverse of f in R_p is

$$F_p(x) = x^6 + 2x^5 + x^3 + x^2 + x + 1 \in R_p.$$

We now compute the public key in the ring R_q:

$$h(x) = F_q(x)g(x) = 20x^6 + 40x^5 + 2x^4 + 38x^3 + 8x^2 + 26x + 30 \in R_q$$

Note that the coefficients of F_q and h range over all of $\mathbb{Z}_q$ even though f and g had only a few small nonzero coefficients. ■

Notice, however, that $F_p(x)f(x) \neq 1$ in $\mathbb{Z}[x]$ — but after you reduce mod $x^7 - 1$ it will be a polynomial whose reduction mod $p = 3$ is 1.

Algorithm 11.6 — NTRU cryptosystem: encryption and decryption. Given (N, p, q, d) and Bob has published his public key h, we proceed as follows.

Encryption: Alice prepares her message as an element $m(x) \in R$ with coefficients chosen only from S_p. She then chooses $r(x) \in T(d,d)$ at random (her ephemeral key) and computes the ciphertext

$$c(x) = pr(x)h(x) + m(x) \in R_q.$$

Decryption: If Bob receives a ciphertext c, he first computes

$$a(x) = f(x)c(x) \in R_q,$$

and then maps this to an element of $\pi_q(a(x)) \in R$ with coefficients in S_q. He recovers the message as

$$m'(x) = F_p(x)\pi_q(a(x)) \in R_p.$$

Let's continue our example before proving the correctness of this algorithm.

■ **Example 11.7** We continue with the example from [HPS14, Example 6.53]. Suppose Alice's message is

$$m(x) = -x^5 + x^3 + x^2 - x + 1.$$

Choose the ephemeral key

$$r(x) = x^6 - x^5 + x - 1.$$

Then the ciphertext is

$$\begin{aligned} c(x) &= ph(x)r(x) + m(x) \mod q, \quad \mod x^N - 1 \\ &= 3h(x)r(x) + m(x) \mod 41, \quad \mod x^7 - 1 \\ &= 3(24x^6 - 48x^5 + 56x^4 - 68x^3 + 54x^2 - 26x + 8) + (-x^5 + x^3 + x^2 - x + 1) \mod 41 \\ &= 31x^6 + 19x^5 + 4x^4 + 2x^3 + 40x^2 + 3x + 25 \mod 41 \end{aligned}$$

which we note disguises the message $m(x)$ since reducing $c(x)$ mod 3 gives $x^6 + x^5 + x^4 - x^3 + x^2 + 1 \neq m(x)$. When Bob receives this message, he computes

$$a(x) = x^6 + 10x^5 + 33x^4 + 40x^3 + 40x^2 + x + 40 \in R_q$$

so

$$\pi_q(a(x)) = x^6 + 10x^5 - 8x^4 - x^3 - x^2 + x - 1 \in R.$$

Then he finds

$$F_p(x)\pi_q(a(x)) = 2x^5 + x^3 + x^2 + 2x + 1 \in R_p,$$

which after applying π_p, is exactly the message we started with. ▪

Proof of correctness. We need to show that $m' = m$. Since

$$c(x) = pr(x)h(x) + m(x) = pr(x)F_q(x)g(x) + m(x) \in R_q$$

we have

$$a(x) = f(x)c(x) = pr(x)(f(x)F_q(x))g(x) + f(x)m(x) = pr(x)g(x) + f(x)m(x) \in R_q.$$

Let us understand this product as an element of R. By construction, each of the polynomials r, g, f have coefficients in $\{0, \pm 1\}$, and those of m are in S_p. When we multiply the two polynomials $r(x) = \sum r_i x^i$ and $g = \sum g_i x^i$ in R, the result is a convolution of the form

$$r(x)g(x) = \sum_{i+j\equiv 0} r_i g_j + \left(\sum_{i+j\equiv 1} r_i g_j\right) x + \cdots + \left(\sum_{i+j\equiv N-1} r_i g_j\right) x^{N-1},$$

where we interpret the sums as being over the set of all pairs of indices (i, j) such that $i + j \equiv k \bmod N$, since $x^{N+k} = x^k$.

Thus the maximum possible value of any coefficient is $2d$ (in absolute value). Similarly, the maximum possible value of any coefficient of $f(x)m(x)$ is bounded by $(2d+1)p/2$. Thus altogether, the coefficients of $pr(x)g(x) + f(x)m(x)$ are, in absolute value, less than

$$p(2d) + (2d+1)p/2 = (3d + \frac{1}{2})p < \frac{q-1}{2}.$$

That is, $\pi_q(a(x)) = pr(x)g(x) + f(x)m(x)$ as computed in R. Therefore we compute

$$\begin{aligned} m'(x) &= F_p(x)\pi_q(a(x)) \in R_p \\ &= p(F_p(x)r(x)g(x)) + (F_p(x)f(x))m(x) \in R_p \\ &= 0 + 1m(x) \in R_p \\ &= m(x) \end{aligned}$$

as required. ■

11.2 Analysis of NTRU

Complexity, briefly: NTRU encryption and decryption are quite fast. Calculating products of polynomials with only $2d$ nonzero coefficients, and those equal to ± 1, amounts to n sums of $2d$ elements of bitsize $\log(q)$. Calculating the coefficients mod q is also fast; the overall complexity is $O(n^2)$.

Security: There are two separate but closely related issues to consider:

- How difficult is it to extract the private keys?
- How difficult is it to extract the message?

We'll focus here on the key extraction problem; if one can solve this then one can certainly extract the message.

Our first observation is that the key is not unique.

Lemma 11.8 Let $f'(x) \in R$ be invertible in R_p and set $g'(x) = \pi_q(f'(x)h(x)) \in R$. If both f' and g' satisfy that all coefficients of

$$pg'(x)\phi(x) + f'(x)m(x)$$

lie in the range $-q/2$ to $q/2$, then the decryption of $m(x)$ will succeed with $f'(x)$ in place of $f(x)$.

Proof. We compute the decryption steps with $f'(x)$; this gives

$$a(x) = f'(x)c(x) = pr(x)f'(x)h(x) + f'(x)m(x) = pr(x)g'(x) + f'(x)m(x) \in R.$$

By hypothesis, this is in the image of π_q. Compute $F'_p(x)$ as the inverse of $f'(x)$ mod p; then

$$F'_p(x)a(x) = pF'_p(x)r(x)g'(x) + F'_p(x)f'(x)m(x) \equiv m(x) \mod p,$$

as required. ■

Note that finding a suitable $f(x)$ is all that is needed to complete both steps of the decryption algorithm.

■ **Example 11.9** If $f_i(x) = x^i f(x) \in R$ for some $i > 0$, then setting $g_i(x) = x^i g(x)$, it follows that

$$(F_i)_q(x)g_i(x) = (x^i f(x))^{-1}(x^i g(x)) = h(x) \mod q,$$

so $f_i(x)h(x) \equiv g(x) \in R_q$. Since all the coefficients of $a(x)$ are the same as if we'd used (f, g), they meet the hypothesis of the lemma, so (f', g') is an alternate key pair. ■

More generally, suppose $\theta(x)$ is any polynomial; then $f'(x) = \theta(x)f(x)$ satisfies

$$f'(x)h(x) = \theta(x)f(x) \star h(x) = \theta(x)g(x) \mod q.$$

Thus, to find an alternate key, one might hope to find a sparse polynomial $\theta(x)$ so that $\theta(x)f(x)$ and $\theta(x)g(x)$ still have small coefficients, in which case $(\theta f, \theta g)$ is another alternate key.

R If we just pick a small polynomial $f'(x)$ at random and compute $f'(x)h(x) \in R_q$ to get $g'(x)$, it is highly unlikely that our resulting polynomial $g'(x)$ will have all coefficients very small (relative to p). (And without knowledge of $f(x)$, that would be our only tactic towards finding a small $\theta'(x)$.)

So: we need to find polynomials $f(x), g(x)$ with small coefficients such that $f(x)h(x) = g(x) \in R_q$. The strongest attacks to NTRU come from turning this into a *shortest vector problem* in a lattice.

11.3 The NTRU Lattice

A lattice is a free $\mathbb{Z}$-module, that is, it is the $\mathbb{Z}$-span of some vectors. If those vectors lie in $\mathbb{R}^m$ or $\mathbb{Z}^m$ then we say it is of full rank if $n = m$. For example, if $\{v_1, v_2, v_3\}$ is a basis for $\mathbb{R}^3$ then

$$L = \{av_1 + bv_2 + cv_3 \mid a, b, c \in \mathbb{Z}\}$$

is a lattice in $\mathbb{R}^3$; it looks like a regularly-spaced set of points, 8 of which are the corners of a parallelepiped. In this chapter, we discuss lattices in $\mathbb{Z}^n$; that is, we only consider bases such that all of the coefficients of the vectors in the basis are integers.

Definition 11.10 — Shortest vector problem. Let $L \subset \mathbb{Z}^n$ be a full rank lattice. The *shortest vector problem* is to find a nonzero vector $v \in L$ such that $\|v\| \le \|w\|$ for all $w \in L \setminus \{0\}$.

For a 2-dimensional lattice, this is quite easy: do Gram-Schmidt orthogonalization on a basis, rounding off your non-integer scalar factors. But as the dimension of the lattice increases, this becomes progressively more difficult. Algorithms that attempt to find a shortest vector, or even just a vector of length “fairly close" to a shortest vector, are called *lattice reduction algorithms*, like LLL and BKZ. For N large and an arbitrary lattice, the SVP is solvable with these only in exponential time.

So how does this relate to NTRU? We first note that if we write a polynomial

$$r(x) = r_0 + r_1x + \cdots + r_{n-1}x^{n-1} \in R$$

as a vector

$$r = [r_0\ r_1\ \cdots\ r_{n-1}]$$

then for any other polynomial $s(x) \in R$, the product $r(x)s(x)$ is the matrix product rS where

$$S = \begin{bmatrix} s_0 & s_1 & \cdots & s_{n-1} \\ s_{n-1} & s_0 & \cdots & s_{n-2} \\ \vdots & \vdots & & \vdots \\ s_1 & s_2 & \cdots & s_0 \end{bmatrix}.$$

Let $h(x)$ be an NTRU public key and write H for its corresponding matrix as above. Then $fH = g \mod q$. That is, $fH = g + qu$ for some polynomial $u(x) \in R$.

We wish to define a lattice whose vectors are such pairs (f, g) (viewed as vectors in $2N$-space), as we’ve established that any suitable short vectors like this will do.

Lemma 11.11 Define

$$M_h := \begin{bmatrix} I_N & H \\ 0 & qI_N \end{bmatrix}$$

where I_N in the $N \times N$ identity matrix. The rows of this matrix are a basis for the NTRU lattice corresponding to $h(x)$. That is, for any $f, u \in \mathbb{Z}^N$, we have

$$[f\ u]M_h = [f\ g]$$

where the pair (f,g) satisfies $f(x)h(x) \equiv g(x) \in R_q$.

Proof. We calculate:

$$[f\ u]M_h = [f\ u]\begin{bmatrix} I_N & H \\ 0 & qI_N \end{bmatrix} = [fI_N + 0\ fH + quI_N] = [f\ g]$$

where

$$fH + qu = g \Leftrightarrow f(x)h(x) = g(x) \in R_q.$$

In particular, since $[f\ u]$ is an integer vector, $[f\ g]$ is an integer vector in the integer span of the rows of M_h. So the rows of M_h span an integral lattice whose short vectors are potential keys for the corresponding NTRU system. ■

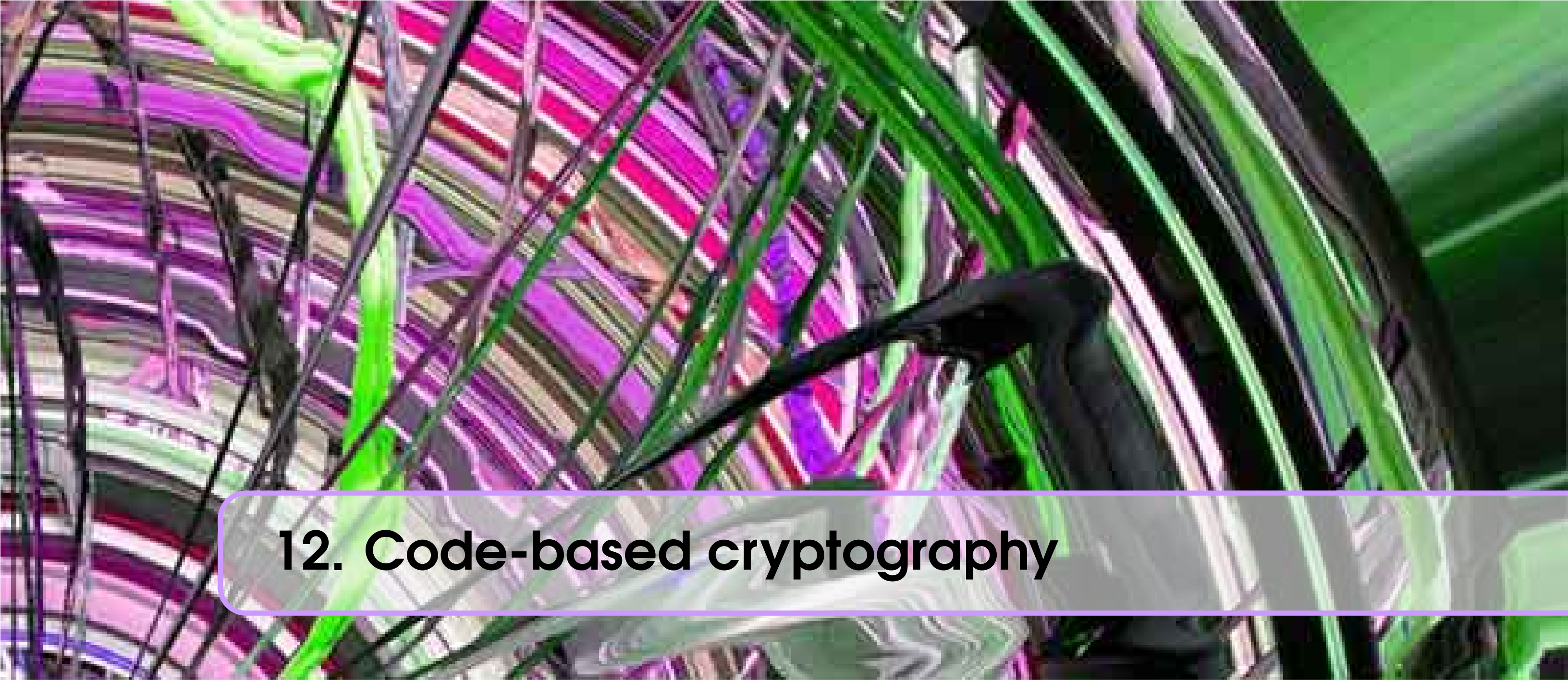
Hence: if a lattice reduction algorithm successfully produces a short vector in the NTRU lattice, then that instance of the NTRU cryptosystem is compromised.

11.4 Conclusions

NTRU has undergone many transformations since its original definition but the ideas at its core remain the same. It continues to be considered a promising and viable cryptographic system, with security related to either the shortest vector problem, or the newer “Ring Learning with Errors" (R-LWE) problem proposed in the 2000s. While NTRU did make it to the 3rd round of submissions for NIST PQC process, it was, however, not selected for standardization by NIST in 2022.

11.5 Exercises

1. Prove that if $f,g \in R_q$ then to compute $fg \in R_q$ we may choose any representatives $f',g' \in \mathbb{Z}[x]$, compute the product $f'(x)g'(x) \in \mathbb{Z}[x]$, and then either first reduce modulo $\langle x^N - 1\rangle$ and then modulo q, or first reduce the coefficients mod q and then reduce the result modulo $\langle x^N - 1\rangle$ in the ring $\mathbb{Z}_q[x]/\langle x^N - 1\rangle$. That is, the following diagram commutes:
$$\begin{tikzcd} \mathbb{Z}[x] \arrow[r, "\text{mod } q"] \arrow[d, "\text{mod } x^N-1"] & \mathbb{Z}_q[x] \arrow[d, "\text{mod } x^N-1"] \\ \mathbb{Z}[x]/\langle x^N-1\rangle \arrow[r, "\text{mod } q"] & \mathbb{Z}_q[x]/\langle x^N-1\rangle. \end{tikzcd}$$
2. In the setting of the example, verify that $f(x)F_p(x) \neq 1$ in R but does reduce to 1 in R_p.
3. In the encryption algorithm, the ciphertext is $c(x) = pr(x)h(x) + m(x)$ in R_q. Why can't Eve simply reduce c mod p to recover m? Consider the example.
4. Prove that if $f \in T(d,d)$ then f cannot be invertible in R_q. Hint: what is $f(1)$?
5. [HPS14, Exercise 6.27] With parameters $N = 7$, $p = 2$ and $q = 37$ (so $S_p = \{0,1\}$ and we use binary instead of trinary coefficients), suppose Bob's private key is $f(x) = x + x^3 + x^6$.
   (i) Compute $F_2(x)$, the inverse of $f(x)$ mod 2, using long division and the extended Euclidean algorithm.
   (ii) Suppose Bob receives the ciphertext $c(x) = 1 + 3x + 3x^2 + 4x^3 + 4x^4 + x^5 + 35x^6$. Decipher the message and find the plaintext.



12. Code-based cryptography

In 1978, Robert McEliece proposes a cryptosystem based on error-correcting codes [McE78]. The one-way trapdoor function at its core is the realization that efficient decoding algorithms for algebraic codes (like BCH, for example) depend on knowing its full structure; but that one can communicate a code by choosing a random basis for it without revealing this structure.

This cryptosystem was not widely adopted because of its (at the time) prohibitively large key sizes. It has withstood some 40 years of scrutiny with very few corrections required, and so is now emerging as a very strong PQC contender.

12.1 McEliece cryptosystem

The error-correcting codes used in McEliece are *Goppa codes*, which derive their structure from algebraic geometry. (The structural data for Goppa codes also uses polynomials and field extensions; a Goppa code is defined by $\Gamma = \{g; \alpha_1, \cdots, \alpha_n\}$ where g is a polynomial in $\mathbb{F}_q$ (with $q = 2^m$) and $\alpha_i \in \mathbb{F}_q$ but are specifically NOT roots of g.) Variants in the literature use certain subcodes of BCH codes, but the cyclic nature of BCH codes has been an exploitable weakness. We will nonetheless consider them as a toy example to understand this cryptosystem.

Suppose that C is an (n,k) BCH code over $\mathbb{F}_q$ of designed distance d, based on a choice of $d-1$ consecutive powers of a primitive nth root of unity, say $\gamma^{\ell+1}, \gamma^{\ell+2}, \cdots, \gamma^{\ell+d-1}$, the least common multiple of whose minimal polynomials over $\mathbb{F}_q$ is a polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $n-k$.

Recall that we can construct a generator matrix G for $C = \langle g(x) \rangle$ by using cyclic shifts of $g(x)$ as its columns; therefore if you know G then you know $g(x)$ and can find the consecutive powers of a primitive nth root of unity γ^j that are its roots — and thus use the efficient decoding algorithm of Section 9.6. Ergo: the first step is to disguise G.

Algorithm 12.1 — McEliece cryptosystem: key generation. Bob chooses a linear (n,k) code over $\mathbb{F}_q$ that corrects $\mathbf{t}$ errors, with an efficient decoding algorithm $D\colon \mathbb{F}_q^n \to \mathbb{F}_q^k$ and corresponding generator matrix G. He then chooses, uniformly at random:

- an invertible $k \times k$ matrix S with entries in $\mathbb{F}_q$; and
- a permutation $\sigma \in \mathscr{S}_n$ with $n \times n$ matrix $P = P_\sigma$.

He computes $\hat{G} = PGS$; this, together with $(n,k,\mathbf{t},q)$, is his public key. The tuple (P,G,S,D) is his private key.

McEliece proposed parameters $(n,k,\mathbf{t},q) \sim (1024,524,50,2)$ in 1978; the NIST proposal parameters include options like the `mceliece460896` : $(4608,3360,96,2^{13})$.

Lemma 12.2 The matrix $\hat{G}$ is a generator matrix for a code equivalent to G that corrects the same number of errors.

Proof. Let C be the code generator by G and $\hat{C}$ the subspace generated by $\hat{G}$, which is thus a code. Since S and P are invertible, it will again be an (n,k) code. Moreover, we have

$$\begin{aligned}\hat{C} &= \{\hat{G}x \mid x \in \mathbb{F}_q^k\}\\ &= \{PGSx \mid x \in \mathbb{F}_q^k\}\\ &= \{PGy \mid y \in \mathbb{F}_q^k\} \quad \text{since } S \text{ is invertible}\\ &= \{Pc \mid c \in C\}\end{aligned}$$

so $\hat{C}$ is the permutation P applied of C, meaning they are equivalent codes.

Since the codes are equivalent, they have the same value of $d_{min}(C)$, and therefore correct the same number of errors. ■

The matrix S is often called a *scrambler matrix* for the role it plays in disguising the (highly structured) matrix G.

Algorithm 12.3 — McEliece cryptosystem: encryption. Alice wants to send a message $m \in \mathbb{F}_q^k$ to Bob. She chooses a random “error vector" $z \in \mathbb{F}_q^n$ of Hamming weight exactly $\mathbf{t}$, and computes her ciphertext as

$$x = \hat{G}m + z.$$

Eve, upon intercepting this message, needs to try to decode x to a nearest codeword c. If she finds c, then she can row reduce $\hat{G}m = c$ to find the message m. Note that it is equally difficult to find c as to find z, so we sometimes say we are looking for the random error vector z. It is infeasible for Eve to use syndrome decoding; the number of syndromes is astronomical (exercise).

Bob, however, knows an efficient decoding algorithm, so has better options.

Algorithm 12.4 — McEliece cryptosystem: decryption. Suppose Bob receives a ciphertext x. Then he decodes x as follows:

1. Compute $x' = P^{-1}x$
2. Decode $m' = D(x')$; then $Gm' = x'$.
3. Compute $m = S^{-1}m'$.

Lemma 12.5 If $x = \hat{G}m + z$ is a valid ciphertext, then Bob's decoding algorithm decodes to the message m.

Proof. Note that the ciphertext can be thought of as the result of a transmission over a noisy channel in which precisly **t** errors occured. If $x = \hat{G}m + z$, then

$$x' = P^{-1}x = P^{-1}(PGS)m + P^{-1}z = G(Sm) + P^{-1}z.$$

Since P is a permutation, $wt(P^{-1}z) = wt(z) = \mathbf{t}$, so Bob's decoding algorithm can correct this error and will return the codeword GSm or, equivalently, the "message" $m' = Sm \in \mathbb{F}_q^k$. Thus $S^{-1}m' = S^{-1}Sm = m$, as required. ■

The security of the McEliece cryptosystem has remained relatively constant since inception, with essentially no significant weaknesses found. Moreover, the best known attack (classical or quantum) uses a technique called information set decoding, the major cost of which is a search. A quantum search algorithm due to Lov Kumar Grover in 1996 reduces the complexity of the search on a list of n terms by a square root, so the rule of thumb is to double the key size to achieve the same level of security against attack.

A downside to the McEliece cryptosystem are the key sizes.

■ **Example 12.6** Suppose $n = 4608$, $k = 3360$, $\mathbf{t} = 96$ and $q = 2^{13}$. The public key is a $n \times k$ matrix over $\mathbb{F}_q$, so requires

$$n \times k \times \log_2(q) = 4608 \times 3360 \times 13 \text{ bits} \approx 24 \text{ Mbytes.}$$

This is a bit extreme, even by today's generous standards! ■

So let's consider some more efficient variants, the ones that were used in the NIST submission.

12.2 Niederreiter cryptosystem

Harald Niederreiter proposed a variant of McEliece in 1986, based on parity check matrices rather than generator matrices. Since a code C with generator matrix G and parity check matrix H can be recovered as either $C = \{Gx \mid x \in \mathbb{F}^k\}$ or $C = \ker(H)$, these two notions are equivalent in terms of the information they convey about the code. Moreover, when computing the syndrome is the key step of the decoding algorithm, it is not necessary to retain the received word at all!

This is true, for example, of BCH codes: the first step was to compute the syndrome relative to a certain parity check matrix, and the entries of this vector (called the syndromes S_i in that algorithm) were the only features of the received word that we used until the final step.

R The parity check matrix H that reveals the efficient decoding algorithm for BCH codes, given in 9.1, has entries in some extension field E of $\mathbb{F}$ of degree ℓ. If you write each element of E as a vector with components in $\mathbb{F}$, then replace each entry of the matrix H with its corresponding vector, then the resulting matrix has entries in $\mathbb{F}$ and has size $(d-1)\ell \times n$. In the following, we'll assume it's actually $(n-k)\times n$; for the true Goppa codes this can be arranged.

Algorithm 12.7 — Niederreiter variant: key generation. Bob chooses a linear (n,k) code over $\mathbb{F}_q$ that corrects $\mathbf{t}$ errors, with an efficient syndrome decoding algorithm and corresponding parity check matrix H. He then chooses, uniformly at random:

- an invertible $(n-k)\times(n-k)$ matrix S with entries in $\mathbb{F}_q$; and
- a permutation $\sigma \in \mathscr{S}_n$ with $n \times n$ matrix P.

He computes $\hat{H} = SHP$; this, together with $(n,k,\mathbf{t},q)$, is his public key. The triple (P,H,S) (and corresponding syndrome decoding algorithm) is his private key.

As before, one can show that the code $\hat{C} = \ker(\hat{H})$ is equivalent to C, so corrects $\mathbf{t}$ errors.

Algorithm 12.8 — Niederreiter variant: encryption. Alice wants to send a message to Bob. She first converts her message to a vector $m \in \mathbb{F}_q^n$ of weight exactly $\mathbf{t}$, and then computes her ciphertext as

$$y = \hat{H}m.$$

In other words, instead of sending her message as a codeword disguised by an error, she turns it into the error, and simply sends the syndrome.

Algorithm 12.9 — Niederreiter variant: decryption. Suppose Bob receives a ciphertext y. He decodes y as follows:

1. Compute $y' = S^{-1}y$.
2. Use syndrome decoding relative to H to find the error m'.
3. Compute $m = P^{-1}m'$.

Let us first ensure that this works.

Lemma 12.10 If $y = \hat{H}m$ is a valid ciphertext, then Bob's decoding algorithm decodes to the message m.

Proof. We have $y' = S^{-1}\hat{H}m = S^{-1}(SHP)m = H(Pm)$. Since P is a permutation matrix, the weight of Pm is equal to that of m, which is $\mathbf{t}$. Thus syndrome decoding recovers $m' = Pm$ from $H(Pm)$, and Bob recovers $m = P^{-1}m' = P^{-1}Pm = m$. ■

What options does Eve have, upon intercepting a ciphertext y? She can set up a table of syndromes, hoping to identify which error gave rise to the syndrome y, but this work is astronomical (see the exercises). In fact, if Eve found some new way to recover m from y, then she could apply this technique to the original McEliece cryptosystem to recover z and hence the message — so the Niederreither variant is no weaker than the original. The advantage it offers is improved key size.

■ **Example 12.11** Suppose $n = 4608$, $k = 3360$, $\mathbf{t} = 96$ and $q = 2^{13}$. The public key is a $(n-k) \times n$ matrix over $\mathbb{F}_q$, so requires

$$(n-k) \times n \times \log_2(q) = 1248 \times 4608 \times 13 \text{ bits} \approx 9 \text{ Mbytes}.$$

Well, that's still huge, but somewhat better. ■

That said, the ciphertexts themselves are short and thus easy to transmit.

■ **Example 12.12** Suppose $n = 4608$, $k = 3360$, $\mathbf{t} = 96$ and $q = 2^{13}$. A ciphertext is a vector in $\mathbb{F}_q^n$, so has size

$$\log_2(q^{n-k}) = 1248 \times 13 \text{ bits} \approx 200 \text{ bytes}.$$

■

12.3 Classic McEliece

The submission to NIST continues the trend to great efficiency.

For example, upon sober second thought: what was the point of the matrix S? It is to scramble the matrix H so that we can't infer the nice form that reveals the algebraic structure of the code. A different way to achieve the same end is to row reduce $\hat{H}$ to its RREF R!

Lemma 12.13 The matrix R is another parity check matrix for $\hat{C}$. We can obtain the syndrome Rm from $\hat{H}m$ and for fixed P and every scrambler matrix S we obtain the same matrix R.

Proof. Recall that row reducing a matrix $\hat{H}$ is the process of multiplying it on the left by various invertible matrices; therefore $R = S'\hat{H}$ for some invertible matrix S'. Given $\hat{H}$ and $y = \hat{H}m$, first row reduce $\hat{H}$ to find S', and then compute Rm as $Rm = S'y$. On the other hand, for every invertible S, the matrix SHP will have the same RREF R. ■

In other words: we lose nothing by omitting S and instead letting row reduction do the scrambling. In practice, Classic McEliece (which is, despite its name, based on the Niederreiter variant) takes this one step further, and omits the permutation matrix P as well (though it offers parameters sets with some permutations); thus the security relies entirely on the decoding problem and the private key is smaller. Again, it is based exclusively on binary Goppa codes but we state it more generally.

Algorithm 12.14 — Classic McEliece: key generation. Bob chooses a linear (n,k) code C over $\mathbb{F}_2$ that corrects $\mathbf{t}$ errors, with an efficient syndrome decoding algorithm and corresponding parity check matrix H, such that the RREF of H is of the form $\hat{H} = [I_{n-k}|T]$ where T has size $(n-k) \times k$. Then T, together with $(n,k,\mathbf{t},2)$, is his public key, and H, together with his syndrome decoding algorithm, is his private key.

Note that Bob's private key is significantly smaller; in fact, he needn't store the matrix H as he can generate it from the data he needs for his efficient decoding algorithm. The public key is also marginally smaller, since we only need to communicate the matrix T.

■ **Example 12.15** Suppose $n = 4608$, $k = 3360$, $\mathbf{t} = 96$ and $q = 2^{13}$. The public key T is a $(n-k) \times k$ matrix over $\mathbb{F}_q$ (which is a $13(n-k) \times k$ matrix over $\mathbb{F}_2$), so requires

$$(n-k) \times k \times \log_2(q) = 1248 \times 1248 \times 13 \text{ bits} \approx 2.5 \text{ Mbytes}.$$

■

Note that Alice can reconstruct $\hat{H}$ from the public key T and so encryption is unchanged from the Niederreiter variant Algorithm 12.8.

Algorithm 12.16 — **Classic McEliece: encryption.** To send a message to Bob, Alice converts it to a vector $m \in \mathbb{F}_q^n$ of weight exactly $\mathbf{t}$ (algorithm provided). She sets $\hat{H} = [I_{n-k}|T]$ and computes her ciphertext as

$$y = \hat{H}m.$$

For decryption, however, Bob can exploit the nature of $\hat{H}$ in a surprising way.

Algorithm 12.17 — **Classic McEliece: decryption.** Suppose Bob receives a ciphertext y. He decodes y as follows:

1. Append k zeros to y to create $x = (y, 0, \cdots, 0) \in \mathbb{F}_q^n$.
2. Decode to find the closest codeword c to x.
3. Set $e = x + c$ and verify that $\mathrm{wt}(e) = \mathbf{t}$ and $He = Hx$.
4. Deduce $m = e$.

This algorithm seems very strange. Let's prove it works.

Lemma 12.18 If Bob receives a valid ciphertext y that is the output of the encryption algorithm with some input m, then his decryption algorithm outputs m.

Proof. Suppose $y = \hat{H}m$ where $\hat{H} = [I_{n-k}|T]$ and m has weight exactly $\mathbf{t}$. Then since $x = (y, 0, \cdots, 0)$ has all nonzero entries in the first $n-k$ coordinates, we have

$$\hat{H}x = [I_{n-k}|T] \begin{bmatrix} y \\ 0 \end{bmatrix} = y + 0 = y = \hat{H}m.$$

Therefore x and m have the same syndrome with respect to $\hat{H}$, which means $x - m \in C$. Therefore they also have the same syndrome with respect to the secret parity check matrix H.

Therefore Bob may apply his decoding algorithm to the received word x to find a codeword $c \in C$ and an error vector $e \in \mathbb{F}_2^n$ such that

$$x = c + e.$$

Then $e = x + c$ since we are working over a field of characteristic 2. If $\mathrm{wt}(e) = \mathbf{t}$ and $He = Hx$ then, since C can correct $\mathbf{t}$ errors, we know this is the unique solution (or coset leader), and thus must be equal to m. ■

In practice, one also takes care in these algorithms to specify what to do if a step fails (as would happen if there were noise on the line, but also if Eve were trying something nefarious).

Even though the operations for encryption and decryption are very efficient (and much simpler than for existing public key cryptosystems), its public key is quite large, and this system was not chosen by NIST for standardization in 2022. It remains a very respected system, with over 40 years of scrutiny and no structural flaws found (when implemented with binary Goppa codes!).

Let's next examine some examples and the main method of attack.

12.4 Information set decoding

Let's consider the original McEliece cryptosystem. We're given a random-looking $n \times k$ matrix G and $x \in F^n$ and trying to find $m \in F^k$ and a vector $e \in F^n$ of Hamming weight at most $\mathbf{t}$ so that the linear system

$$Gm = x - e$$

is consistent, meaning that $x - e \in C$.

To get a feel for this, let's consider an example.

■ **Example 12.19** Suppose that e is nonzero only in its first and ith coordinates; then in matrix form this system looks like

$$\begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,k} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ g_{i,1} & g_{i,2} & \cdots & g_{i,k} \\ g_{i+1,1} & g_{i+1,2} & \cdots & g_{i+1,k} \\ g_{i+2,1} & g_{i+2,2} & \cdots & g_{i+2,k} \\ \vdots & \vdots & \vdots & \vdots \\ g_{n-1,1} & g_{n-1,2} & \cdots & g_{n-1,k} \\ g_{n,1} & g_{n,2} & \cdots & g_{n,k} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ x_{i+1} \\ x_{i+2} \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

So: If we just removed the first and ith rows of both G and x, then the new linear system $G'm = x' - e' = x'$ would be consistent (and easy to solve!).

The problem: we don't know beforehand which rows we need to remove... ■

This example is in fact our plan of attack. Let's set some notation.

Definition 12.20 Let G be an $n \times k$ generator matrix for a code C and $x \in F^n$. For any subset $S \subset \{1, 2, \cdots, n\}$, let G_S denote the $|S| \times k$ matrix obtained from G by taking just the rows indexed by elements that are in S, and similary let $x_S \in F^{|S|}$ be the vector

$$x_S = (x_{i_1}, x_{i_2}, \cdots, x_{i_{|S|}}) \quad \text{where} \quad S = \{i_1, \cdots, i_{|S|}\} \text{ and } i_1 < i_2 < \cdots < i_{|S|}.$$

If G_S is invertible then this set S is called an *information set*.

Let Ω_S be the matrix obtained from the $n \times n$ identity matrix I_n by keeping only those rows corresponding to S. Then $G_S = \Omega_S G$ and $x_S = \Omega_S x$. If $S = \{i_1, i_2, \cdots, i_{|S|}\}$ and $i_1 < i_2 < \cdots < i_{|S|}$ then we can define

Ω_S explicitly as the $|S| \times n$ matrix with entries

$$(\Omega_S)_{k,\ell} = \begin{cases} 1 & \text{if } \ell = i_k; \\ 0 & \text{otherwise.} \end{cases}$$

■ **Example 12.21** Consider $n = 5$ and $S = \{1, 2, 4\}$. Then

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

Thus for example if x is a vector $abcde$ we have

$$\Omega_S \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} a \\ b \\ d \end{bmatrix} = x_S.$$

■

The following algorithm is due to Eugene Prange [Pra62].

Algorithm 12.22 — Information Set Decoding. Let C be an (n,k) code that corrects $\mathbf{t}$ errors. Given a received word $x \in F^n$ that is the result of at most $\mathbf{t}$ errors, we find the closest codeword c to x as follows:

```
1. Choose a random subset $S \subset \{1, 2, \cdots, n\}$ of size $k$.
2. If $G_S$ is invertible, then
    (a) set $e = x - G G_S^{-1} x_S$; and
    (b) if $\mathrm{wt}(e) \leq \mathbf{t}$, then stop and return $e$;
3. Return to Step 1 (that is, if you did not find an answer at Step 2(b).)
```

That is, one keeps choosing size-k information sets until one finds the answer.

Note that although Step 2(a) is phrased with G_S^{-1}, it is in fact equivalent to solving $G_S u = x_S$ and then computing Gu, which can usually be done much more efficiently.

Let's do an example to illustrate the steps before proving that it always works.

■ **Example 12.23** Consider the binary Hamming code C (see Section 5.5) with parity check and generator matrices

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad \text{and} \quad G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

We know that C corrects one error, so set $\mathbf{t} = 1$. We have $k = 4$. Suppose we receive $x = 1010010$. Since $Hx = 110 \neq 0$ we know there was an error. Remembering syndrome decoding for Hamming codes, we know exactly where that error is — but let's forget that and instead apply information set decoding.

First choose $S = \{1,2,3,4\}$. Then $G_S = I_4$, which is certainly invertible, so we compute

$$e = x - GG_S^{-1}x_S = x - GI_4x_S = x - G\begin{bmatrix}1\\0\\1\\0\end{bmatrix} = 1010010 - 1010100 = 0000110.$$

Since $wt(e) > \mathbf{t}$, this fails and we try again.

Now choose $S = \{4,5,6,7\}$. Then we have

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

We could decide if it is invertible by calculating its determinant, but since our next step is to solve $u = G_S^{-1}x_S$ anyway, we can just row reduce to solve $G_Su = x_S$— and if it turns out G_S is not invertible, we just stop. Thus

$$[G_S|x_S] = \begin{bmatrix}0&0&0&1&|&0\\1&1&0&1&|&0\\1&1&1&0&|&1\\1&0&1&1&|&0\end{bmatrix} \sim \begin{bmatrix}1&0&0&0&|&1\\0&1&0&0&|&1\\0&0&1&0&|&1\\0&0&0&1&|&0\end{bmatrix}$$

which shows both that G_S was invertible (was this an accident? see Lemma 12.24, below) and that $G_S^{-1}x_S = u = 1110$. We compute

$$e = x - Gu = 1010010 - 1110010 = 0100000,$$

which indeed has weight $wt(e) \leq \mathbf{t}$. Therefore this is was the error vector (as expected) and our codeword was 1110010 and our message was 1110. ■

Now let us show that the information set decoding algorithm always terminates, and always on a correct solution. We do this as a sequence of lemmas.

Lemma 12.24 If Algorithm 12.22 terminates and outputs e, then $e \in F^n$ and $x - e$ is the closest codeword in C to x. That is, the algorithm will never produce an incorrect answer.

Proof. First note that G is of size $n \times k$, G_S is of size $k \times k$, and $x_S \in F^k$, so if G_S is invertible then the product $GG_S^{-1}x_S$ is well-defined and thus $e \in F^n$.

Suppose $e = x - GG_S^{-1}x_S$ is a vector of weight at most $\mathbf{t}$. Set $m = G_S^{-1}x_S \in F^k$; then $c = Gm \in C$ by definition, and since $\mathrm{wt}(x - c) \leq \mathbf{t}$, and C is $\mathbf{t}$-error-correcting, we conclude that $c = GG_S^{-1}x_S$ is indeed the closest codeword to x. ■

In particular, this proof showed us that if S is an information set such that (b) holds, then the closest codeword will be

$$c = GG_S^{-1}x_S \in C.$$

Moreover, since G is an injective map from F^k to F^n, it means that when S is successful, we have

$$m = G_S^{-1}x_S.$$

This is part of a more general observation: given any information set S of size k, if $Gm = c$ then $m = G_S^{-1}c_S$. That is, this is one way to find m once you know c; you don't need to row reduce the entire overdetermined linear system to solve for m.

■ **Example 12.25** Suppose G is in standard form

$$G = \begin{bmatrix} I_k \\ A \end{bmatrix},$$

and $S = \{1, 2, \cdots, k\}$. Then $G_S = I_k$ is invertible, so S is an information set for a systematic code. In this case,

$$c = GG_S^{-1}x_S = Gx_S = \begin{bmatrix} x_S \\ Ax_S \end{bmatrix},$$

which will be the correct codeword if and only if $x_S = m$. ■

Let's now understand the conditions in Step 2, towards deciding if this algorithm will actually always terminate.

Lemma 12.26 Let C be a linear (n,k) code with $d_{min}(C) = d$ and generator matrix G. Then there exists a size-k information set S. If $k = n-d+1$ then every subset of $\{1, 2, \cdots, n\}$ of size k is an information set but if $k < n-d+1$ then this does not hold.

By the Singleton bound we have $k \le n-d+1$, so this says it is only for MDS codes that every $k \times k$ submatrix of G is invertible.

Proof. The matrix G has rank k, so the rank of G^T is also k. Row reduce G^T to RREF and let $S = \{i_1, i_2, \cdots, i_k\}$ denote the indices of the columns containing leading ones. Then G_S is a matrix whose transpose row reduces to the identity matrix, so is invertible.

Now let S be an arbitrary set of k indices. Recall that a square matrix is invertible iff its nullspace is just the zero space. So suppose that $m \in \ker(G_S)$ is nonzero. Then $c = Gm$ is also nonzero (since G is an injective map). However,

$$G_S m = (\Omega_S G)m = \Omega_S(Gm) = (Gm)_S = c_S,$$

which is 0 since $m \in \ker(G_S)$. Therefore $0 < \mathrm{wt}(c) \le n - |S| = n-k$.

Thus, if C is a code such that $d_{min} \ge n-k+1$, this can never happen and every set of size k is an information set. By the Singleton bound, we know $k \le n - d_{min} + 1$; therefore this condition on C is actually equality and we are in the setting of MDS codes.

Conversely, if we have a nonzero codeword c of weight less than or equal to $n-k$ (so: not an MDS code), then choosing S among the indices where the coordinate $c_i = 0$ yields $c_S = 0$ so a matrix G_S that is not invertible. ■

Thus if the matrix G_S is not invertible then the set S (or the corresponding rows of G) does not have enough information to reconstruct the codeword — the set S does not identify enough "information."

Lemma 12.27 Suppose that $\overline{e}$ is the error vector (of weight at most $\mathbf{t}$) corresponding to the input x of the algorithm, and that S is an information set for C. Then Step 2 succeeds, with output $e = \overline{e}$, if and only if S is disjoint from the positions of the errors, that is, if and only if $\overline{e}_S = 0$.

Proof. Note that

$$(GG_S^{-1}x_S)_S = \Omega_S GG_S^{-1}x_S = G_S G_S^{-1}x_S = x_S,$$

so if $e = x - GG_S^{-1}x_S$, then $e_S = x_S - x_S = 0$. Therefore if the algorithm returns e (which implies by Lemma 12.24 that it is the correct answer $\overline{e}$) then necessarily $e_S = 0$. Conversely, if $\overline{e}_S = 0$ and $x = c + \overline{e}$ then $x_S = c_S$ and therefore $GG_S^{-1}x_S = c$. Thus the algorithm outputs $e = \overline{e}$. ■

Finally: why must the algorithm terminate?

Lemma 12.28 For any vector $e \in F^n$ such that $\mathrm{wt}(e) \leq \mathbf{t}$, there exists an information set S of size k such that $e_S = 0$.

Proof. Suppose not. This would imply that there exists a vector $e \in F^n$ such that $\mathrm{wt}(e) \leq \mathbf{t}$ and for all information sets S of size k, $e_S \neq 0$. Let T be the set of indices corresponding to the coordinates where e is zero. Then the hypothesis implies that G_T has no $k \times k$ invertible submatrices, which by the process of row reduction, implies that the rank of G_T is less than k so the kernel of G_T has dimension at least 1. Let $m \in \ker(G_T)$ be a nonzero element of the kernel; then $\mathrm{wt}(Gm) \leq n - |T| = \mathrm{wt}(e)$, which is a contradiction since $\mathrm{wt}(e) \leq \mathbf{t} < d_{min}(C)$. ■

12.5 Attacking McEliece

To apply the information set decoding algorithm to attack Classic McEliece, we construct a generator matrix

$$\hat{G} = \begin{bmatrix} -T \\ I_k \end{bmatrix}$$

and use the word $x = (y, 0, \cdots, 0) \in \mathbb{F}_q^n$ that we construct as the first step of the decryption algorithm (which is not secret).

How long will it take this algorithm to crack the message?

Suppose we choose a set S of size k. There are $\binom{n}{\mathbf{t}}$ possible subsets of the error positions of e and of those possibilities, $\binom{n-k}{\mathbf{t}}$ of them correspond to options for which the error positions all lie outside of S. So given an information set, the odds of success are about $\binom{n-k}{\mathbf{t}} / \binom{n}{\mathbf{t}}$.

■ **Example 12.29** If $n = 4608$, $k = 3360$ and $\mathbf{t} = 96$, then this yields[1]

$$\binom{n}{\mathbf{t}} = \binom{4608}{96} \approx 1.853769798 \times 10^{201}$$

whereas

$$\binom{n-k}{\mathbf{t}} = \binom{1248}{96} \approx 4.085867957 \times 10^{145}$$

Therefore, given S, the odds that it does not intersect the error positions in e is to the order of $10^{145-201} = 10^{-56}$. ■

On the other hand, the odds that S will give rise to an invertible matrix G_S are much higher. Supposing that our public key is indistinguishable from a random $n \times k$ matrix, then we can estimate this probability as the probability that a randomly generated $k \times k$ matrix is invertible. Taking $q = 2$ for simplicity, this probability is approximately

$$a(k) = (1 - \frac{1}{2})(1 - \frac{1}{4})(1 - \frac{1}{8}) \cdots (1 - \frac{1}{2^k})$$

(see exercises). Taking the limit as $k \to \infty$, we get[2]

$$\lim_{k \to \infty} a(k) \approx 0.2887880950866024212788997219294585937270.$$

Thus, for k sufficiently large, the chance of a random subset of indices being an information set is about 29%.

Thus the probability of success on each iteration of the Information Set Decoding algorithm is simply about $0.29\binom{n-k}{\mathbf{t}}/\binom{n}{\mathbf{t}}$; another way of saying it is that we expect to have to do about

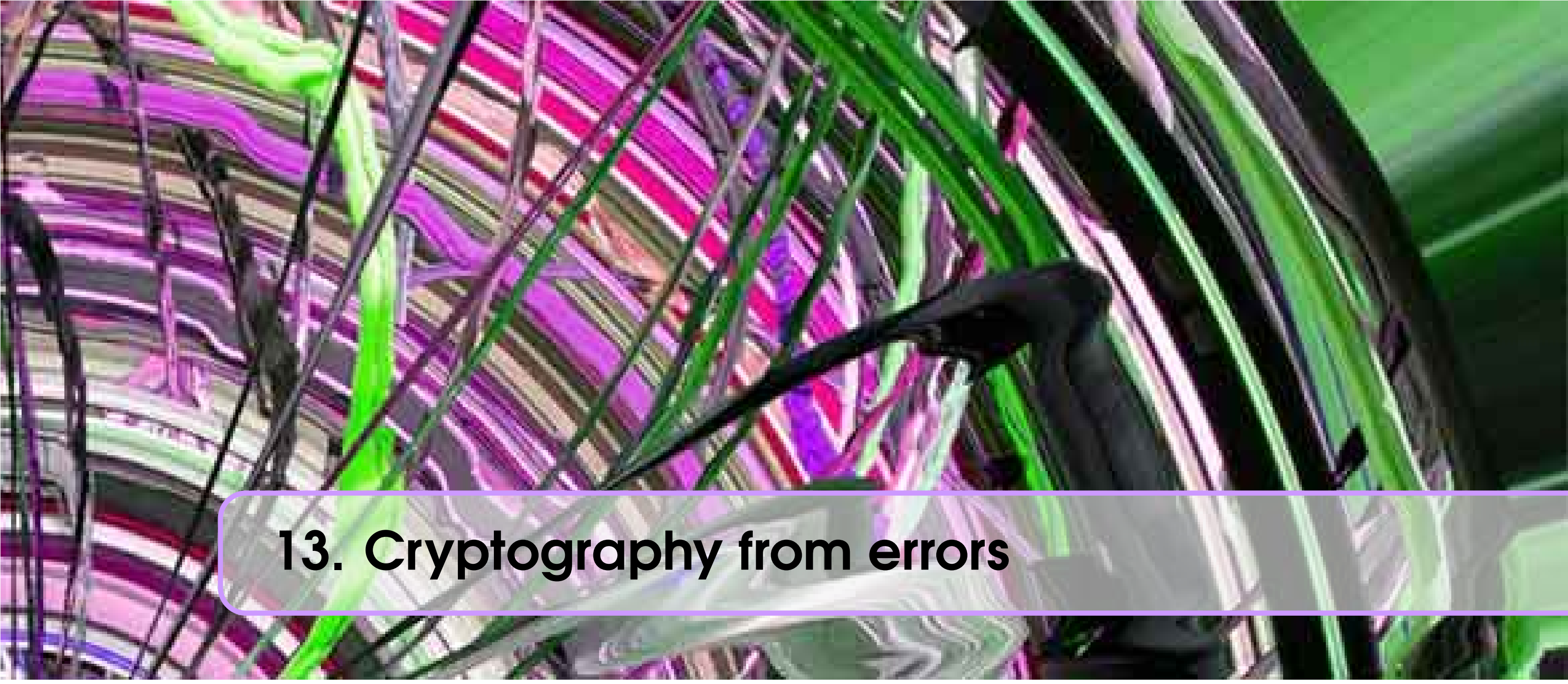
$$\frac{\binom{n}{\mathbf{t}}}{0.29\binom{n-k}{\mathbf{t}}}$$

iterations of the algorithm before succeeding. In our example above this comes out to approximately 2^{187}.

Information set decoding is the most effective attack known against Classic McEliece.

[1]using an online calculator https://www.calculatorsoup.com/calculators/discretemathematics/combinations.php whose accuracy I cannot verify!

[2]This is from the Online Encyclopaedia of Integer Sequences



13. Cryptography from errors

Why is decoding so difficult? After all, if we receive $x = Gm + e$ for some small error vector e, then this is almost a linear system; if $e = 0$ we could solve it with row reduction. The issue: the system $Gm = x$ doesn't have a solution — it's an inconsistent linear system, and all the known techniques for solving linear systems do not give any information about approximate solutions.

This problem, in one form or another, is at the core of several post-quantum cryptosystems (including NTRU). This connection was established in a paper by Oded Regev in 2009 [Reg09], where he defined LWE and showed that if there were an efficient algorithm to solve it, then there would be an efficient algorithm to decode random linear codes, and an efficient *quantum* algorithm to solve the shortest vector problem. LWE can be defined as follows.

13.1 Learning with Errors (LWE)

Note: For LWE, we use boldface letters $\mathbf{e}$ for vectors, and the convention that e_i is the ith coordinate vector of $\mathbf{e}$. If A is a matrix, then $\mathbf{a}_i$ denotes the ith row of A. We will interpret vectors as row or column vectors as convenient.

Problem 13.1 — **Learning with errors.** Setup: Let $F = \mathbb{Z}_p$ and $m > n > 0$ be integers. We choose an $m \times n$ matrix A, and a vector $\mathbf{s} \in F^n$, uniformly at random. Let $\chi\colon F \to \mathbb{R}$ be a probability distribution (centred at 0) for errors, and generate an error vector $\mathbf{e} \in F^m$ by sampling $e_1, \cdots, e_m$ from this distribution. Let

$$\mathbf{b} = A\mathbf{s} + \mathbf{e} \in F^m.$$

Problem: given A and $\mathbf{b}$, determine $\mathbf{s}$.

But wait, you say. In linear algebra, we learned how to find the least-squares best solution to an inconsistent system: to approximately solve the inconsistent system $Ax = b$, project b orthogonally onto the column space of A, giving a nearby vector c, and then solve $Ax = c$. By [GJN21, Theorem 20.2.2], this comes down to solving the consistent linear system

$$A^T Ax = A^T b.$$

Why can't we apply that here? You'll explore this in the exercises, but in a nutshell: we do not have a well-defined notion of orthogonal projection onto a subspace when working over the integers, or over a finite field.

Regev also proposed a public key cryptosystem based on LWE [Reg09, §5]. To set it up, we need to be more precise about the probability distribution χ in the statement of LWE.

Algorithm 13.2 — LWE: specifications. Choose $n > 1$ and p a prime number between n^2 and $2n^2$. Set $F = \mathbb{Z}_p$ and choose m such that $2^m > p^{n+1}$. Recall from our NTRU chapter that we defined

$$S_p = \{-\frac{p-1}{2}, \cdots, -2, -1, 0, 1, 2, \cdots, \frac{p-1}{2}\}.$$

Choose a value $\delta > 0$ and an "error distribution" $\chi \colon S_p \to \mathbb{R}$ for which the following holds: for any $1 \leq k \leq m$, the sum E of k values sampled from χ satisfies

$$-\frac{p-1}{4} \leq E \leq \frac{p-1}{4}$$

with probability at least $1 - \delta$.

■ **Example 13.3** Suppose $n = 2$; then $p = 5$ satisfies $n^2 \leq p \leq 2n^2$, so we choose $F = \mathbb{Z}_5 = \{-2, -1, 0, 1, 2\}$. Let's take $m = 7$, then $2^m = 128 > 125 = p^3$. Suppose for simplicity we choose the probability distribution $\chi_\ell \colon S_5 \to \mathbb{R}$ given by

$$\chi(n) = \begin{cases} \frac{1}{\ell} & \text{if } n = 1 \\ \frac{\ell-1}{\ell} & \text{if } n = 0 \\ 0 & \text{otherwise,} \end{cases}$$

where $\ell > 2$. Suppose we independently sample $(e_i)_{i=1}^k$ from this distribution and set $E = \sum e_i$. In this case, $\frac{p-1}{4} = 1$, so we want to know with what probability we have $-1 \leq E \leq 1$. Well, $Pr(E = -1) = 0$,

$$Pr(E = 0) = \prod_{i=1}^{k} \frac{\ell - 1}{\ell} \qquad \text{and} \qquad Pr(E = 1) = \frac{k}{\ell} \prod_{i=1}^{k-1} \frac{\ell - 1}{\ell}.$$

So for example if $\ell = 32$ then $Pr(|E| \leq 1) < 0.99$ so we can choose $\delta = 0.01$. In practice, you might choose your error tolerance δ first and solve for ℓ! ■

Algorithm 13.4 — LWE: key generation. Following the LWE specifications, create the datum $(A, \mathbf{s}, \mathbf{b})$ as in the setup to the LWE problem. Define Bob's private key to be $\mathbf{s}$. Define the public key to be the pair $(A, \mathbf{b})$.

■ **Example 13.5** Continuing as above, we might generate

$$A = \begin{bmatrix} 1 & 2 \\ 2 & -1 \\ 0 & 1 \\ 1 & -1 \\ 2 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = A\mathbf{s} + \mathbf{e} = \begin{bmatrix} -2 \\ 1 \\ 2 \\ -2 \\ -2 \\ 1 \\ 0 \end{bmatrix}$$

where A and $\mathbf{s}$ were chosen uniformly at random, but we have sampled the coordinates of $\mathbf{e}$ from χ as in the preceding example.

Then $\mathbf{s} = (-1, 2)$ is Bob's private key, while the public key is the pair consisting of the matrix A and the vector $\mathbf{b}$. ■

We see that we could apply the principles of information set decoding to try to recover $\mathbf{s}$ from the public key. To do so we would replace Step 2(b) with "if the sum of the coordinates of e, viewed as elements of S_p, is less than $(p-1)/4$ in absolute value, then stop and return e". This would succeed whenever the error vector $\mathbf{e}$ has at least n coordinates equal to 0 (and the corresponding row vectors of A are linearly independent).

There are a few ways to make this attack infeasible: one is to choose p sufficiently large so that we can make most terms nonzero; another (following our analysis in the previous section) is to choose n and m sufficiently large so that the probability of avoiding errors is vanishingly small.

Algorithm 13.6 — **LWE: encryption.** To encrypt a bit $\alpha \in \{0,1\}$, first choose a random subset $S \subset \{1, 2, \cdots, m\}$, and define $\mathbf{1}_S$ as the *indicator vector* of S, that is, it is the **row** vector with m coordinates 0 and 1 such that the ith coordinate is 1 if and only if $i \in S$. Then ciphertext is a pair $(\mathbf{a}, b)$ where

$$\mathbf{a} = \mathbf{1}_S A, \qquad b = \frac{p-1}{2}\alpha + \mathbf{1}_S \mathbf{b}.$$

R One can also think of $\mathbf{a}$ as $\sum_{i \in S} \mathbf{a}_i$; then we similary have $\mathbf{1}_S \mathbf{b} = \sum_{i \in S} b_i$.

One way for Eve to find α from $(\mathbf{a}, b)$ would be to try to guess S, and thus recreate the sum. Thinking of A as the transpose of a matrix H reveals that this is, in the binary case, the same problem as brute-force syndrome decoding (finding a linear combination of the columns of H that yield $\mathbf{a}$).

On the other hand, Bob doesn't need to discover S at all.

Algorithm 13.7 — **LWE: decryption.** If Bob receives $(\mathbf{a}, b)$, then he uses his secret key $\mathbf{s}$ to compute

$$b - \mathbf{a} \cdot \mathbf{s}$$

where $\cdot$ denotes the dot product. If the answer is closer to 0 than to $\frac{p-1}{2}$, he decrypts to $\alpha = 0$, otherwise, he decrypts to $\alpha = 1$.

This time, our decryption is not guaranteed to work, but we have some control over the possibility of decryption failure. Note that a value $x \in F$, viewed as an element of S_p, will be closer to 0 than to $(p-1)/2$ exactly when $-\frac{p-1}{4} \leq x \leq \frac{p-1}{4}$.

Lemma 13.8 — Regev, Lemma 5.1. With parameters as above, upon receipt of an encryption of the bit α, the decryption algorithm returns α with probability at least $1-\delta$.

Proof. Recall that $\mathbf{b} = A\mathbf{s} + \mathbf{e}$, where $\mathbf{e}$ was a small error vector with coefficients distributed according to χ. Suppose the secret set is S, so that $\mathbf{a} = \mathbf{1}_S A$ (as a row vector) and $b = \mathbf{1}_S\mathbf{b} + \frac{p-1}{2}\alpha$. Then Bob's computation can be written as

$$\begin{aligned} x = b - \mathbf{a}\cdot\mathbf{s} &= (\mathbf{1}_S\mathbf{b} + \frac{p-1}{2}\alpha) - \mathbf{1}_S A\mathbf{s} \\ &= \mathbf{1}_S(\mathbf{b} - A\mathbf{s}) + \frac{p-1}{2}\alpha \\ &= \mathbf{1}_S\mathbf{e} + \frac{p-1}{2}\alpha. \end{aligned}$$

Since $\mathbf{1}_S\mathbf{e} = \sum_{i\in S} e_i$ and $|S| \leq m$, our hypothesis on $\mathbf{e}$ ensures that $|\mathbf{1}_S\mathbf{e}| \leq \frac{p-1}{4}$ (viewed as elements of S_p) with probability at least $1-\delta$, in which case x will round down to 0 if $\alpha = 0$ and to 1 otherwise. ■

■ **Example 13.9** With our public key as before, Alice chooses $S = \{1,4,6\}$ and creates the vector $\mathbf{1}_S = 1001010$ and computes

$$\mathbf{a} = \mathbf{1}_S A = \begin{bmatrix} -2 & 2 \end{bmatrix}, \quad \mathbf{1}_S\mathbf{b} = 2.$$

Then since $(p-1)/2 = 2$, she encrypts the bit $\alpha = 1$ as

$$(\mathbf{a}, b) = (\mathbf{a}, \mathbf{1}_S\mathbf{b} + 2) = ((-2,2), -1).$$

Bob, upon receiving this pair, computes

$$b - \mathbf{a}\cdot\mathbf{s} = -1 - (-2,2)\cdot(-1,2) = -2$$

and, deciding that this is closer to 2 than to 0, decrypts $\alpha = 1$.

Our example is too simple to foil Eve, though! See the exercises. ■

Well, this LWE cryptosystem is nice and all — but that's a lot of work to encrypt a single bit! The value in this system is that it provides a proof of concept of using the LWE problem — a problem that, roughly speaking, if solvable in an efficient way (classical or quantum) would essentially provide an efficient solution for all lattice-based and code-based cryptosystems.

13.2 Crystals Kyber

For our final cryptosystem, we describe (with minor simplifications) one of the algorithms selected for standardization by NIST in 2022, called Kyber [ABD+21]. It is part of the CRYSTALS (CRYptographic

SuiTe for Algebraic LatticeS) [ABD+22] collection of cryptographic algorithms, that include a signature scheme called Dilithium.[1] It is based on ideas arising from NTRU, McEliece, and above all, LWE.

As with Classic McEliece, and the Diffie–Hellman key exchange, it is a Key Encapsulation Mechanism (KEM), meaning that rather than encrypting a message *per se*, it generates a secret key that Alice and Bob subsequently share.

Choose n a power of 2 and a prime q such that $n|(q-1)$. In practice, they propose $n = 256$ and $q = 3329$. Choose a small integer k (say, $k \in \{2,3,4\}$). Define

$$R = \mathbb{Z}[x]/\langle x^n + 1\rangle \quad \text{and} \quad R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle.$$

Note that these are NOT the same rings used in NTRU — the ideal is generated in this case by $x^n + 1$. (Since $n = 2^8$, it is known that this is a cyclotomic polynomial, which is irreducible over $\mathbb{Q}$ but in general will factor over $\mathbb{Z}_q$ into a product of irreducible factors of the same degree.)

We specify a noise distribution χ_η that is a centred binomial distribution taking values between $-\eta$ and η, where $\eta \in \{2,3\}$.

Algorithm 13.10 — Kyber: Key Generation. Generate a random $k \times k$ matrix A with entries in R_q, and choose two vectors $\mathbf{s}, \mathbf{e} \in R_q^k$ whose entries are polynomials with coefficients sampled from χ. The private key is $\mathbf{s}$; the public key is the pair $(A, \mathbf{b} = A\mathbf{s} + \mathbf{e})$.

In other words, we follow the same format as for LWE, with the difference that A is a square matrix, and the entries come from the ring R_q rather than $\mathbb{Z}_2$.

■ **Example 13.11** For a toy example, let's take $n = 2$ and $q = 11$, with $k = 2$. Then we could generate

$$A = \begin{bmatrix} -2+4x & 3-2x \\ 5x & -3+x \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} x \\ 1+x \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} 0 \\ x \end{bmatrix}.$$

Then (remembering that we are working mod 11 and mod x^2+1 so that $x^2 = -1$:

$$\mathbf{b} = A\mathbf{s} + \mathbf{e} = \begin{bmatrix} -2+4x & 3-2x \\ 5x & -3+x \end{bmatrix} \begin{bmatrix} x \\ 1+x \end{bmatrix} + \begin{bmatrix} 0 \\ x \end{bmatrix} = \begin{bmatrix} -2x+4x^2+(3-2x)(1+x)+0 \\ 5x^2+(-3+x)(1+x)+x \end{bmatrix}$$
$$= \begin{bmatrix} -2x-4+3+3x-2x+2 \\ -5-3-2x-1+x \end{bmatrix} = \begin{bmatrix} 1-x \\ 2-x \end{bmatrix}.$$

Our public keys are A and $\mathbf{b}$ but $\mathbf{s}$ is our private key. ■

Algorithm 13.12 — Kyber: Encryption. To encrypt a message from Z_2^n to Bob, Alice expresses it as a polynomial $m \in R_q$ with binary coefficients. She then samples vectors $\mathbf{r}, \mathbf{f} \in R_q^k$ and an element $e_0 \in R_q$ according to the distribution χ. Her ciphertext is the pair $(\mathbf{u}, v)$ that she computes as:

$$\mathbf{u} = A^T\mathbf{r} + \mathbf{f} \in R_q^k \quad \text{and} \quad v = \mathbf{b}^T\mathbf{r} + e_0 + \frac{q-1}{2}m \in R_q.$$

[1] All Star Wars and Star Trek fans will know the origins of the names of these two algorithms!

Note that if $\mathbf{r} = \mathbf{1}_S$ then $A^T\mathbf{r}$ and $\mathbf{b}^T\mathbf{r} + \frac{q-1}{2}m$ would be precisely the analogue of LWE (generalizing from $\mathbb{Z}_2$ to R_q) — see the exercises. However, for added security there is an additional noise term added to each of the parts of the ciphertext.

■ **Example 13.13** Continuing with our example above, suppose we wish to encrypt $m(x) = x \in R_q$. Since $k = 2$ we could choose

$$\mathbf{r} = \begin{bmatrix} 1 \\ -x \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} x \\ 0 \end{bmatrix}, \quad e_0 = -1.$$

Then we have

$$\mathbf{u} = A^T\mathbf{r} + \mathbf{f} = \begin{bmatrix} -2+4x & 5x \\ 3-2x & -3+x \end{bmatrix} \begin{bmatrix} 1 \\ -x \end{bmatrix} + \begin{bmatrix} x \\ 0 \end{bmatrix} = \begin{bmatrix} -2+4x-5x^2+x \\ 3-2x+3x-x^2+0 \end{bmatrix} = \begin{bmatrix} 3+5x \\ 4+x \end{bmatrix}$$

and

$$\begin{aligned} v &= \mathbf{b}^T\mathbf{r} + e_0 + \frac{q-1}{2}m \\ &= (1-x, 2-x)\cdot(1,-x) + (-1) + 5(x) \\ &= 1 - x - 2x + x^2 - 1 + 5x \\ &= -1 + 2x \end{aligned}$$

■

Algorithm 13.14 — Kyber: Decryption. Given the ciphertext $(\mathbf{u}, v)$ and his private key $\mathbf{s}$, Bob computes

$$y = v - \mathbf{s}^T\mathbf{u} \in R_q,$$

and writes his answer with coefficients from the set S_q. He then decodes each coefficient as a 0 if it lies between $-\frac{q-1}{4}$ and $\frac{q-1}{4}$ and as a 1 if not, thus recovering a binary polynomial m.

■ **Example 13.15** If we receive the ciphertext $\mathbf{u} = (3+5x, 4+x)$ and $v = -1+2x$, then with our private key $(x, 1+x)$ we compute

$$\begin{aligned} y &= v - \mathbf{s}^T\mathbf{u} \\ &= -1 + 2x - (x, 1+x)\cdot(3+5x, 4+x) \\ &= -1 + 2x - (3x + 5x^2 + 4 + 5x + x^2) \\ &= -1 + 2x - 8x - 6x^2 - 4 \\ &= 1 + 5x. \end{aligned}$$

Since $1 \in [-10/4, 10/4]$ but 5 is not, we decipher $0 + x$, which was our original message. ■

The parameters for Kyber are chosen so as to ensure that the probability of decryption failure is $\delta \approx 2^{-139}$, that is, in practical terms Bob will always recover the message.

What we see is that Kyber combines the fundamental problem of LWE and code-based cryptography with the efficient and excellent shuffling properties of polynomial multiplication (as used in NTRU) to make a fairly compact encryption system! There is much more to explore about Kyber — but for now, the expectation is that it will be a good bet for the post-quantum world.

13.3 Exercises

1. Bob needs to generate a random $k \times k$ binary matrix that is invertible. How many elements are there in $M_k(\mathbb{Z}_2)$ and how many of them are invertible? Generate a random 5×5 binary matrix.
2. Prove that the set of matrices $\{GS \mid S \in \mathrm{GL}_k(\mathbb{F})\}$ is exactly the set of *all* generator matrices for C. (Here, $\mathrm{GL}_k(\mathbb{F})$ denotes the set of all invertible $k \times k$ matrices with entries in $\mathbb{F}$.)
3. Discuss the effect of choosing a random invertible $n \times n$ matrix P in place of a permutation matrix in the key generation step.
4. For the McEliece cryptosystem, and for the Niederreiter variant: How much storage space is required for the private and public keys corresponding to a binary $(1024, 524)$ code C? Compare with RSA, where the public and private keys may each be up to 4 Kilobytes.
5. Suppose Eve intercepts the ciphertext and decides to use syndrome decoding to recover m. How many syndromes are there to compute, if we use a $(1024, 524)$ binary code that can correct $\mathbf{t} = 50$ errors?
6. Prove that the code $\hat{C} = \ker(\hat{H})$ in the Niederreiter variant is equivalent to Bob's secret code C.
7. The "message" in classic McEliece is a vector $m \in F^n$ of weight exactly $\mathbf{t}$. Would m be a good choice for a random key to use in a subsequent symmetric cipher?
8. Describe an algorithm to generate a binary vector of weight $\mathbf{t}$ and length n, such that each such vector is equally likely to occur.
9. Can you think of an algorithm to encode a message as a weight-$\mathbf{t}$ vector in $\mathbb{F}_q^n$ for some q?
10. Let C be the $(3, 1)$ binary repetition code, with generator matrix G. Suppose we receive the word $y = 110$. Show that the linear system $G^T Gm = G^T y$ has a unique solution but this solution does not give the closest codeword.
11. Let A be an $n \times k$ matrix of rank $k < n$. Show that $\ker(A^T) = (\mathrm{Col}(A))^{\perp}$. Show that if $\mathrm{Col}(A) \cap \mathrm{Col}(A)^{\perp} = \{0\}$, then $A^T A$ is invertible.
12. Give an example of a code C such that $C^{\perp} \cap C \neq \{0\}$. Let G be a generator matrix for C. Find an example of a vector y such that neither $Gm = y$ nor $G^T Gm = G^T y$ has a solution.
13. We say that a vector $v \in F^n$ has an orthogonal projection onto a subspace W if there exists $w \in W$ and $w' \in F^n$ such that $v = w + w'$ and $w \cdot w' = 0$. Give examples of one-dimensional subspaces W and W' of $\mathbb{Z}_2^5$ such that every vector has an orthogonal projection onto W but such that there exist vectors that have no orthogonal projection onto W'.
14. In $\mathbb{R}^n$, every vector has an orthogonal projection onto every subspace W. Let $w \in W$ be the projection of v; show that the minimum value of $\{\|v - u\| : u \in W\}$ is attained for $u = w$. Now examine your proof of this fact and identify the part(s) that fail when we replace $\mathbb{R}$ with a finite field F and $\|x\|$ with the Hamming weight $\mathrm{wt}(x)$.
15. For Goppa codes of the size used in Classic McEliece, it is known that about 29% of information sets S of size k will yield an invertible matrix G_S. If the code has length n and dimension k, and there are $\mathbf{t}$ errors in the message, what is a formula for the percentage of information sets will successfully avoid the error locations? Can you estimate this value for the given parameter sets?

13.4 Final thoughts

Algebraic structures are the ideal framework for computer applications: the objects are often efficiently enumerable, come in infinite variety, and are amenable to rigourous analysis. Where are the frontiers of research today?

With LDPC and turbo codes, engineers are able to use communication channels at very close to capacity. Structured codes, like BCH, are valuable for applications, like cryptography, where one needs to be able to analyse an entire scalable class of codes, and prove various properties of the code and of its decoding algorithm. As such, there is still room to advance the subject; in recent years, the study of Goppa codes has led the way.

Public-key cryptographic protocols rely on algebraic structures. Some examples of subject areas that are widely used in cryptographic applications, but that we did not discuss, include lattices (modules over an integral domain) and systems of multivariate polynomials over a finite field. As these cryptosystems rely on advanced mathematical structures (such as the ones you've learned in this course), the cryptographic community capable of analysing and implementing them is relatively small. In the coming years, as the new generation of PQC is broadly implemented, ring theory will become a plus on one's CV!

Finally, the advent of quantum computers brings its own fascinating collection of challenges — from quantum error correction to finding sets of easy-to-implement unitary matrices that generate large groups — challenges that are solved with tools from algebra, graph theory and lots of advanced linear algebra.

Enjoy!

IV Appendix

A. Mathematical background

Let's now introduce some of the key mathematical constructions we'll need to define, analyse and construct codes.

A.1 The Euclidean algorithm and Extended Euclidean Algorithm

The Euclidean algorithm first appeared the book *Elements*[1] by the Greek mathematician Euclid, circa 300 BCE. It computes the greatest common divisor (gcd) of two positive integers so efficiently that it is still basically the algorithm implemented on computers today.

Recall that if a,b are integers with $a \neq 0$ then we say a divides b if there exists a quotient $q \in \mathbb{Z}$ such that $b = qa$. When a does not divide b, there is a unique quotient q and remainder r satisfying $0 \leq r < a$ such that $b = qa + r$.

We state it for integers, but if you have read Section 6.3 then you realize a variant of it applies in any ring where we have a replacement for the notion of size.

Algorithm A.1 — Euclidean algorithm. Given two positive integers $a \leq b$, the following algorithm outputs $\gcd(a,b)$:

1. Perform division with remainder to get $b = qa + r$ where q is the quotient and r is the remainder; thus $0 \leq r < a$.
2. If $r = 0$, then output a, and we are done.
3. If $r > 0$, then repeat step 1 with the pair (r,a) in place of (a,b).

■ **Example A.2** To find $\gcd(144,30)$ we do

Step 1 $144 = 4 \times 30 + 24$

[1]See http://www.physics.ntua.gr/~mourmouras/euclid/index.html or a more modern take such as https://archive.org/details/euclid-elements-redux_201809/euclid-a4/mode/2up.

Step 2 No, not done yet
Step 3 Now calculate $\gcd(24,30)$
Step 1 $30 = 1 \times 24 + 6$
Step 2 No, not done yet
Step 3 Now calculate $\gcd(6,24)$
Step 1 $24 = 4 \times 6 + 0$
Step 2 Yes, $\gcd(24,6) = 6$.
So the algorithm says the answer is $\gcd(144,30) = 6$. ■

But why is this true? Let's prove that the algorithm outputs $\gcd(a,b)$.

Proof. Suppose $a \leq b$. Then the algorithm starts with $r_0 = b$ and $r_1 = a$, and in each iteration produces a new remainder r_i. Since these remainders are a decreasing sequence of nonnegative numbers, the algorithm must terminate after finitely many steps. So suppose the sequence of remainders is $r_0, r_1, r_2, \cdots, r_n, 0$, so that the output of the algorithm is r_n.

We first claim that r_n divides all r_i with $i \leq n$. We proceed by backward induction Since $r_{n-1} = qr_n$ with no remainder, r_n divides r_{n-1} (and it also divides itself). Suppose r_n divides $r_n, r_{n-1}, \cdots, r_i$. Then since $r_{i-1} = qr_i + r_{i+1}$ for some quotient q, r_n divides the right hand side by the induction hypothesis and thus also the left side, so r_n divides r_{i-1}. Thus by induction, r_n divides $r_0 = b$ and $r_1 = a$.

We need to show it is the greatest common divisor, by showing that every common divisor of a and b also divides r_n. Suppose s divides both a and b. Then since $b = qa + r_2$, we deduce that s divides r_2. In fact, if s divides r_i and r_{i+1}, then since $r_{i+1} - qr_i = r_{i-1}$ for some quotient q, s must divide the right hand side as well. Thus by induction, we infer that s divides r_n. Hence $r_n = \gcd(a,b)$. ■

One can show that the number of steps needed for this algorithm to terminate on a given pair of inputs (a,b) is proportional on average to $\max\{\ln(a), \ln(b)\}$. Thus even for very large inputs it converges quickly.

A consequence of the Euclidean algorithm is that you can always express $\gcd(a,b)$ as an integer linear combination of a and b!

Theorem A.3 — Generalized Euclidean Algorithm. Let $a \leq b$ be positive integers and let $d = \gcd(a,b)$. Then there exist $s,t \in \mathbb{Z}$ such that

$$d = sa + tb,$$

and moreover these coefficients can be calculated from the steps of the Euclidean algorithm.

We illustrate with an example.

■ **Example A.4** Consider the calculation that produced $\gcd(144,30) = 6$ in Example A.2. We had

$$\begin{aligned} 144 &= 4 \times 30 + 24 \\ 30 &= 1 \times 24 + 6 \\ 24 &= 4 \times 6 \end{aligned}$$

Let's solve for 6, starting from the second-last equation and working upwards, writing $a = 30$ and $b = 144$ when they occur to help us keep track of what we're doing:

$$\begin{aligned}6 &= 30 - 1 \times 24\\ &= a - 1 \times 24\\ &= a - 1 \times (144 - 4 \times 30)\\ &= a - (b - 4a)\\ &= 5a - b.\end{aligned}$$

Check: $5 \times 30 - 144 = 6$. So $s = 5$ and $t = -1$. ▪

Let's prove the theorem.

Proof. Let us reprise our notation from the proof of the Euclidean algorithm and set $r_0 = b$ and $r_1 = a$, so that the steps of the Euclidean algorithm are

$$\begin{aligned}r_0 &= q_1 r_1 + r_2 & &\Longrightarrow r_2 = b - q_1 a\\ r_1 &= q_2 r_2 + r_3 & &\Longrightarrow r_3 = a - q^2 r_2\\ &\vdots & &\qquad\vdots\\ r_{i-2} &= q_{i-1} r_{i-1} + r_i & &\Longrightarrow r_i = r_{i-2} - q_{i-1} r_{i-1}\\ &\vdots & &\qquad\vdots\\ r_{n-2} &= q_{n-1} r_{n-1} + r_n & &\Longrightarrow r_n = r_{n-2} - q_{n-1} r_{n-1}\\ r_{n-1} &= q_n r_n + 0. & &\end{aligned}$$

Since each r_i is expressed as a linear combination of r_j with $j < i$, we can proceed backwards through the list of equations on the right, substituting the appropriate equation for each occurrence of r_j with $2 \leq j < n$, until r_n is expressed just in terms of a and b, as required. ■

A.2 Working in base n

We usually work with decimal numbers, that is, numbers in base 10. This expresses itself by our shorthand notation:

$$534 = 5 \times 10^2 + 3 \times 10^1 + 4 \times 10^0,$$

where the coefficients in front of each power of 10 in this expansion are chosen to lie in the set $\{0, 1, \cdots, 9\}$. We needn't have chosen 10; for example, the ancient Mayans independently developed the place-number system in mathematics (in fact, a few centuries earlier than its discovery in Asia), using base 20, and therefore 20 distinct coefficient symbols[2].

Thus for example, if we use base 2, where our coefficients are just $\{0, 1\}$, we can write

$$110110_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 32 + 16 + 4 + 2 = 54$$

[2]Check them out on the Wiki page: https://en.wikipedia.org/wiki/Maya_numerals.

where the subscript 2 lets us know that this is shorthand for the *binary* expansion of a number (although we omit it when there is no possibility of confusion). Another convenient base is 16, which gives rise to *hexadecimal numbers*; in this case the coefficient symbols are denoted $\{0,1,\cdots,9,A,B,C,D,E,F\}$ and some common notation includes, for example,

$$0x6F := 6F_{16} = 6\times 16 + F = 96 + 15 = 111.$$

To state the following lemma, recall that the *ceiling function* is defined on $y\in\mathbb{R}$ by $\lceil y\rceil = \min\{k\in\mathbb{Z} \mid k\geq y\}$ and the *floor function* is defined on $y\in\mathbb{R}$ by $\lfloor y\rfloor = \max\{k\in\mathbb{Z}\mid k\leq y\}$.

Lemma A.5 Let x,n be integers with $n>1$ and $x>0$. Set $k=\lfloor \log_n(x)\rfloor$. Then there exist unique integers $a_i\in\{0,1,\cdots,n-1\}$, for $i\in\{0,\cdots,k\}$, such that

$$x = a_k n^k + a_{k-1}n^{k-1} + \cdots + a_1 n + a_0. \tag{A.1}$$

Proof. We begin by noting that $x=1$ has a unique expansion of the form (A.1): if any a_i with $i>0$ were nonzero, then the sum of these positive numbers would be greater than n, and $n>1$ by hypothesis; and taking $a_i=0$ for $i>0$ and $a_0=1$ yields x.

Suppose now that we have shown the existence and uniqueness of the expression (A.1) for all positive integers less than x. Set $k=\lfloor\log_n(x)\rfloor$. Then $k\leq\log_n(x)<k+1$ so since $n>1$ we conclude $n^k\leq x<n^{k+1}$. Since

$$n^k = 1\times n^k < 2\times n^k < \cdots < (n-1)\times n^k < n\times n^k = n^{k+1},$$

there is a unique $a_k\in\{1,\cdots,n-1\}$ such that $a_kn^k\leq x<(a_k+1)n^k$. Set $\bar{x}=x-a_kn^k$ and note that this is less than both x and n^k by construction.

Therefore by induction, $\bar{x}$ has a unique base n expansion $\bar{x}=a_{k-1}n^{k-1}+\cdots+a_1n+a_0$, where the coefficient of a_{k-1} may be zero and all coefficients lie in $\{0,1,\cdots,n-1\}$. The lemma follows. ■

The proof of the lemma gives an effective algorithm for computing the base n expansion of any positive integer.

A.3 Working mod n

Let $n>1$ and define addition and multiplication on the set $\mathbb{Z}_n=\{0,1,\cdots,n-1\}$ by the rule: first add or multiply as integers, and then take the remainder upon division by n.

■ **Example A.6** In $\mathbb{Z}_5$, we have the identities

$$2+2=4,\quad 2+3=0,\quad 2+4=1,\quad ,\cdots$$

and

$$1\times 3=3,\quad 2\times 3=1,\quad 3\times 3=4,\quad ,\cdots.$$

■

■ **Example A.7** In the *binary field* $\mathbb{Z}_2$, the operations on $\{0,1\}$ are those of boolean arithmetic. That is, we have

$$a+b=a\vee b, \quad ab=a\wedge b.$$

So $1+1=0$ but all other sums and products are as for $\mathbb{Z}$. In computer science, we often write $a\oplus b$ for this sum operation, to distinguish it from the ordinary sum. ■

A more elegant explanation of modular arithmetic is via equivalence classes.

Definition A.8 Let $n>1$. Two elements $a,b\in\mathbb{Z}$ are said to be *equivalent mod n* if there exists an integer k such that $a-b=kn$.

The proof of the following result is a lovely exercise with equivalence relations, and is left to the reader.

Proposition A.9 Equivalence mod n defines and equivalence relation on $\mathbb{Z}$, whose classes are represented by $\mathbb{Z}_n$. Addition and multiplication in $\mathbb{Z}$ induce well-defined operations on these equivalence classes that coincide with taking remainders mod n.

In this context, we write $\mathbb{Z}/n\mathbb{Z}$ for the set of equivalence classes.

Addition and multiplication in $\mathbb{Z}_n$ satisfy several useful and familiar properties:

A1 For all a,b we have $a+b=b+a$ (commutativity of addition);
A2 For all a,b,c we have $a+(b+c)=(a+b)+c$ (associativity of addition);
A3 There exists an element $b\in\mathbb{Z}_n$ such that for all a we have $a+b=a$ (existence of additive identity) which we henceforth denote 0;
A4 For all a there is an element $b\in\mathbb{Z}_n$ such that $a+b=0$ (existence of an additive inverse) which we henceforth denote $-a$;
M1 For all a,b we have $ab=ba$ (commutativity of multiplication);
M2 For all a,b,c we have $a(bc)=(ab)c$ (associativity of multiplication);
M3 There exists an element $b\in\mathbb{Z}_n$ such that for all a we have $ab=a$ (existence of multiplicative identity) which we henceforth denote 1;
D1 For all a,b,c we have $a(b+c)=ab+ac$ (distributivity of multiplication over addition).

Any set G with an operation $+\colon G\times G\to G$ satisfying A1-A4 is called an additive (abelian) group (see Section 2.1); sets with two operations satisfying all these properties are called (commutative unital) rings (see Chapter 6).

What about inverses? It doesn't make sense to ask if $\dfrac{1}{2}$ is an element of $\mathbb{Z}_5$ — $\mathbb{Z}_5$ consists of equivalence classes of integers, not rational numbers. Instead, we ask: does $\mathbb{Z}_5$ contain an element b that *acts like* $\frac{1}{2}$, in that

$$2b=1$$

with respect to the multiplication in $\mathbb{Z}_5$? The answer is yes: take $b=3$. We call 3 the multiplicative inverse of 2 mod 5.

Theorem A.10 Let $n>1$. Then for any $a\in\mathbb{Z}$, a has a multiplicative inverse mod n if and only if $\gcd(a,n)=1$.

Proof. Let $d = \gcd(a,n)$. Then by the generalized Euclidean algorithm we can solve for integers s,t such that

$$d = sa + tn.$$

Considering this equation modulo n, we have $sa = d \bmod n$. Therefore if $d = 1$ then a is invertible.

Conversely, if $d > 1$ then it is a factor of both a and n. Since it is a factor of n, we see directly that for any integer k,

$$dk \in \{d, 2d, \cdots, (n/d-1)d, n\} \mod n,$$

that is, the cycle repeats and does not include 1. Since a is a multiple of d, any multiple of a is a multiple of d, and none of these multiples is congruent to 1 mod n. ■

A.4 Permutations

Let T be a finite set.

Definition A.11 A *permutation* of T is a bijection from T to itself. The set of all permutations of a set of cardinality n is denoted $\mathscr{S}_n$.

Note that $|\mathscr{S}_n| = n!$.

Suppose $T = \{1, 2, \cdots, n\}$. Let us write our permutations with *cycle notation*. A cycle

$$\sigma = (a_1 a_2 \cdots a_{k-1} a_k)$$

(where $k \le n$, and the a_i are distinct elements of T) means that

$$\sigma(a_1) = a_2, \quad \sigma(a_2) = a_3, \quad \cdots \quad \sigma(a_{k-1} = a_k, \quad \sigma(a_k) = a_1,$$

and that $\sigma(b) = b$ for any element $b \in T \setminus \{a_1, \cdots, a_k\}$.

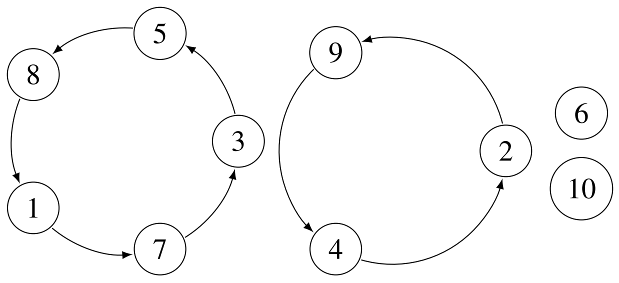


Figure A.1: The action of the permutation $\sigma = (1\ 7\ 3\ 5\ 8)(2\ 9\ 4)$ on $T = \{1, 2, \cdots, 9, 10\}$. By their omission, $\sigma(6) = 6$ and $\sigma(10) = 10$.

Not every permutation is a cycle, but every permutation can be written as a product of disjoint cycles (meaning: the sets of points that they move are disjoint), as in Figure A.1.

Lemma A.12 The set of permutations, with composition, forms a (nonabelian) group.

Proof. Composing two permutations of T is again a permutation of T (perhaps a trivial one!), so $\mathscr{S}_n$ is closed under composition. Composition of functions is associative. (How does one check? Let $f,g,h\colon T \to T$. To compare $f\circ(g\circ h)$ with $(f\circ g)\circ h$ we have to apply both sides to an element $x\in T$ and be sure the answer is the same. Recall that $(a\circ b)(x) = a(b(x))$. So we have

$$(f\circ(g\circ h))(x) = f((g\circ h)(x)) = f(g(h(x)))$$

and

$$((f\circ g)\circ h)(x) = (f\circ g)(h(x)) = f(g(h(x))).$$

So that's the same.) Composition of function is not usually commutative!

The identity element is the function $e\colon T\to T$ defined by $e(x) = x$ for all x. This is a permutation (a trivial one). The inverse of a permutation is obtained by reversing all the arrows of the cycles; this is again a permutation. Thus $\mathscr{S}_n$ is a group.

To see that it is nonabelian in general, note that

$$(12)\circ(23) = (123) \quad \text{but} \quad (23)\circ(12) = (132)$$

so $\mathscr{S}_n$ is nonabelian for $n \geq 3$. ■

Permutations can also act by linear transformations, as follows. Define a map $\varphi\colon \mathscr{S}_n \to GL(F^n) = \{g\in M_{n\times n}(F) \mid \det(g)\neq 0\}$ that takes $\sigma\in\mathscr{S}_n$ to $\varphi(\sigma) = P_\sigma$, where P_σ is the linear transformation defined on the standard basis of F^n by

$$P_\sigma(e_i) = e_{\sigma(i)} \quad \text{for all } i = 1,2,\cdots,n.$$

The matrix of P_σ is obtained from that of the identity by permuting the columns, so has exactly one 1 in each row and column. It is called a permutation matrix, and its determinant is either 1 or -1.

This permutation action can seem counterintuitive, as in the following lemma.

Lemma A.13 Let $v = (v_1, v_2, \cdots, v_n)\in F^n$ and let $\sigma\in\mathscr{S}_n$. Then

$$P_\sigma(v_1,\cdots,v_n) = (v_{\sigma^{-1}(1)}, v_{\sigma^{-1}(2)}, \cdots, v_{\sigma^{-1}(n)}).$$

Proof. Let $\{e_1,\cdots,e_n\}$ denote the standard basis of F^n. Then $v = (v_1,v_2,\cdots,v_n)$ is shorthand for

$$v = v_1e_1 + \cdots + v_ne_n.$$

Therefore

$$\begin{aligned} P_\sigma(v) &= P_\sigma(v_1e_1+\cdots+v_ne_n)\\ &= v_1P_\sigma(e_1)+\cdots+v_nP_\sigma(e_n)\\ &= v_1e_{\sigma(1)}+\cdots+v_ne_{\sigma(n)}. \end{aligned}$$

Now to rewrite this as a coordinate vector, we have to put the basis vectors back in order. Note that if $\sigma(i) = j$ then $\sigma^{-1}(j) = i$, and so

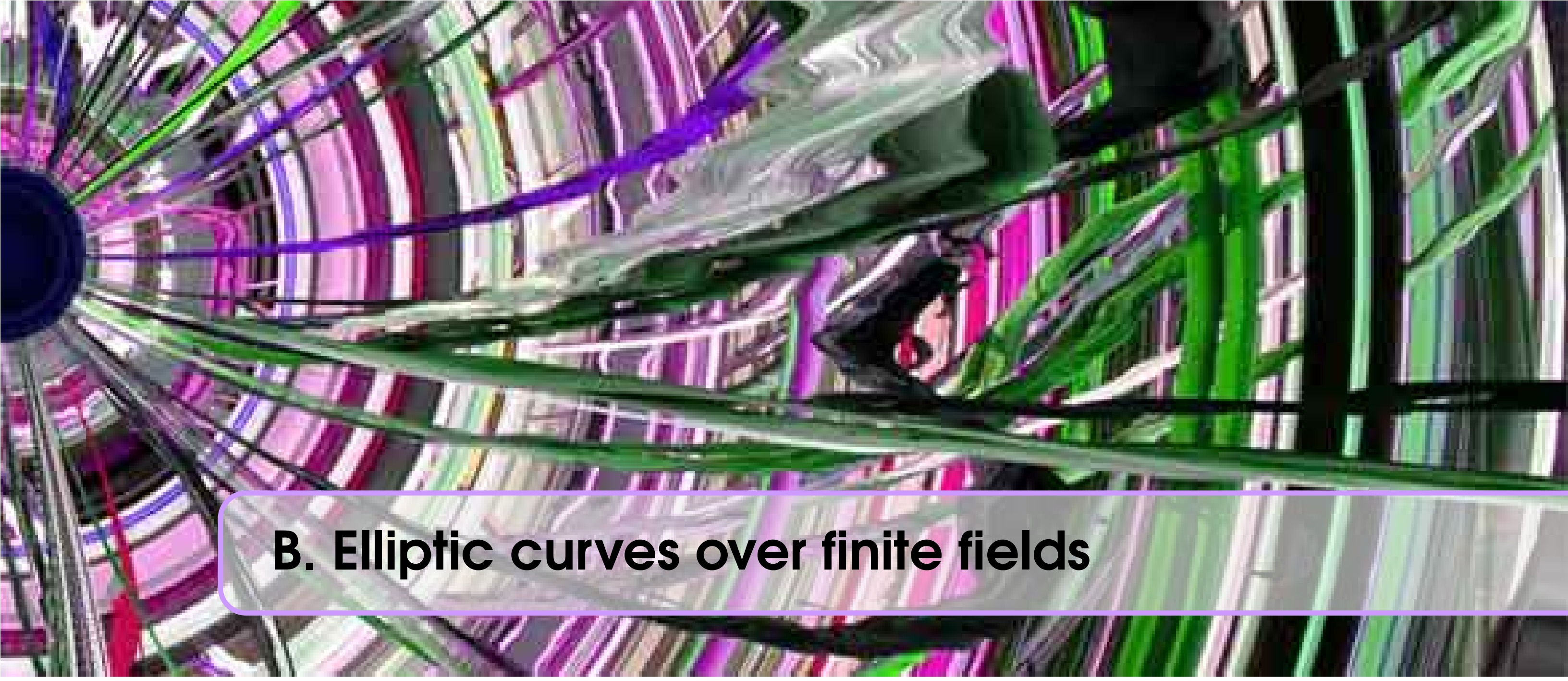
$$v_ie_{\sigma(i)} = v_ie_j = v_{\sigma^{-1}(j)}e_j.$$

The result follows. ■

We sometimes write $\sigma(v)$ instead of $P_\sigma(v)$.

A.5 Exercises

1. Apply the extended Euclidean algorithm to find x, y such that $ax + by = 1$, where $a = 4$ and $b = 13$.
2. Use the Euclidean algorithm to find $\gcd(1056, 249)$. Then solve for this gcd as a linear combination of 1056 and 249 with integer coefficients.
3. The algorithm converges most slowly if all of the quotients are 1. Construct an example of a pair (a, b) such that the Euclidean algorithm takes 8 steps to converge to the answer.
4. Compute 157 in base 2, base 3 and base 150. Is it necessary for the base to be a prime number?
5. What decimal numbers are represented by 101010_2, 12311_5 and 24_{16}?
6. Define an and .
7. Prove Proposition A.9.
8. Compute the following mod 17: $12 + 8$, 12×8, $8 - 12$.
9. Solve $12x = 8 \bmod 17$.
10. Find all invertible elements (with respect to multiplication) in $\mathbb{Z}_{10}$.
11. Use the Euclidean algorithm to find the inverse of 230 modulo 1201.
12. Consider the permutation of $T = \{1, 2, 3, 4, 5\}$ that sends $1 \mapsto 2$, $2 \mapsto 5$, $3 \mapsto 4$, $4 \mapsto 3$, $5 \mapsto 1$. Write this as a product of disjoint cycles.
13. Find the permutation matrix corresponding to $\sigma = (12)(35) \in \mathscr{S}_5$.



B. Elliptic curves over finite fields

We learned in Chapter 10 that finite fields are an easy and very natural source of cyclic groups; unfortunately, the discrete logarithm problem on finite fields, particularly on fields of order 2^k for some $k > 0$, can be solved in a reasonable amount of time on a classical computer (at least, in any case nice enough for anyone to try to implement ElGamal in the first place). (This saga was played out in the 1980s, particularly by researchers in Waterloo.)

Elliptic curves were proposed in the 1980s for cryptographic purposes, and are where fields of the form $\mathbb{F}_{2^k}$ are being used in cryptography today.

B.1 Definitions

Let K be a field and let $\overline{K}$ denote an algebraic closure of K. (Recall that an algebraically closed field is characterised by the property that every polynomial with coefficients in $\overline{K}$ factors completely into linear factors over $\overline{K}$.) If $K = \mathbb{F}_q$ is the finite field with q elements (where $q = p^r$ for some $r \geq 1$ and p prime) then the algebraic closure of K is

$$\overline{K} = \bigcup_{m \geq 1} \mathbb{F}_{q^m}.$$

Definition B.1 The *projective plane* $P^2(K)$ over K is the set of all lines in K^3; that is, it is the set of equivalence classes of the relation $\sim$ on $K^3 \setminus \{(0,0,0)\}$, where $(X,Y,Z) \sim (X',Y',Z')$ if there is a scalar $\lambda \in K$ such that $X' = \lambda X$, $Y' = \lambda Y$ and $Z' = \lambda Z$. We write $(X : Y : Z)$ for the equivalence class of (X,Y,Z).

A *homogeneous polynomial in several variables* is $F \in K[X_1, X_2, \cdots, X_n]$ is one in which each term has the same total degree, where the total degree of $X_1^{i_1} X_2^{i_2} \cdots X_n^{i_n}$ is $i_1 + i_2 + \cdots + i_n$. So $X_1 X_2 + X_2^2$ is homogeneous but $X_1 + X_1^2$ is not.

Given a homogeneous polynomial over K in three variables, such as

$$F(X,Y,Z) = Y^2Z - X^3 + XZ^2$$

we consider its *zero set*

$$\{(X,Y,Z) \mid F(X,Y,Z) = 0\}.$$

Since F is homogeneous, $F(X,Y,Z) = 0$ if and only if $F(\lambda X, \lambda Y, \lambda Z) = 0$ (for any $\lambda \neq 0$). So in fact, we can think of the zero set (excluding $(0,0,0)$) as a subset of $P^2(K)$ and set

$$E(K) = \{(X:Y:Z) \in P^2(K) \mid F(X,Y,Z) = 0\}.$$

We could equally consider the larger set

$$E(\overline{K}) = \{(X:Y:Z) \in P^2(\overline{K}) \mid F(X,Y,Z) = 0\}.$$

We say that E is an *algebraic variety* (which we identify with the set $E(\overline{K})$) and the set $E(K)$ is its set of K-rational points.

■ **Example B.2** If $K = \mathbb{F}_3$ and F is as above, then

$$E(K) = \{(0:1:0), (1:0:1), (2:0:1), (0:0:1)\}$$

whereas there are infinitely many points on $E(\overline{K})$. ■

The main idea of algebraic geometry is to look at curves and surfaces which are defined as zero sets of polynomials. Here, we are using projective geometry, and so restrict our attention to zero sets of homogeneous polynomials. The homogeneity of the polynomial implies that if a point is a zero of the polynomial, then so is any scalar multiple of that point; in other words, the surface defined as the zero set of this polynomial is composed entirely of straight lines through the origin.

This isn't the easiest way to think of these zero sets, or of projective space, though. Since projective $P^2(\mathbb{R})$ is the set of all lines in $\mathbb{R}^3$, then if we consider just the points of unit length in $\mathbb{R}^3$, we'll have representatives for every equivalence class in $P^2(\mathbb{R})$ and these lie on the unit sphere. More accurately, we have exactly two representatives for each class, given by a point on the sphere and its negative (which lies exactly opposite, like the north and south pole do). If we were to just take the top half of the sphere, then most of our equivalence classes would reduce to a single point instead of two points; but the points on the equator are still doubly-represented. We think of this as: a very large part (a big open subset) of $P^2(\mathbb{R})$ looks just like regular Euclidean $\mathbb{R}^2$ (by doing a projection from the origin through the top half of the sphere onto the plane $z = 1$, for example); but then there's a bit more.

Definition B.3 An *elliptic curve* is the zero set E in $P^2(\overline{K})$ of a nonsingular polynomial of the form

$$F(X,Y,Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3,$$

where $a_1, a_2, a_3, a_4, a_5 \in K$, where nonsingular (sometimes: smooth) means that for all $(x:y:z) \in E$, at least one of the partial derivatives

$$\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}$$

is nonzero at the point (x,y,z).

We note that $\mathcal{O} := (0:1:0) \in E$ and that this is the *only* point $(X:Y:Z)$ in E for which $Z = 0$. Hence, on $E \setminus \{\mathcal{O}\}$ we can make the change of variables

$$x = X/Z, y = Y/Z$$

and then it makes sense to say

$$E = \{\mathcal{O}\} \cup \{(x,y) \in \overline{K}^2 \mid y^2 + a_1xy + a_3y = x^3 + a^2x^2 + a_4x + a_6\}.$$

We say this is an expression for E (or for F) in *affine coordinates* and we call $\mathcal{O}$ the *point at infinity* of E.

R How this makes sense geometrically: the only point of the elliptic curve which lies on the equator is the point $\mathcal{O}$; so if we just take the portion of our elliptic curve which lies on the top half of the sphere (plus this extra point we need to keep track of), then the projection discussed above is realized algebraically by this change of coordinates. "Most" of our elliptic curve (including its most interesting features) takes place in the top half of the sphere, and by making this change of coordinates we're moving from projective geometry to Euclidean geometry, which is much simpler to draw and think about.

By making further changes of variables, we can simplify the above general expression even more. Namely, every elliptic curve can be represented in affine coordinates as the point at infinity together with all solutions in $\overline{K}^2$ of:

- $y^2 = x^3 + ax + b$, where the RHS has no multiple roots, if $p \neq 2,3$
- $y^2 = x^3 + ax^2 + bx + c$, where the RHS has no multiple roots, if $p = 3$;
- either $y^2 + cy = x^3 + ax + b$ or $y^2 + xy = x^3 + ax^2 + b$ (no condition) if $p = 2$.

Let's concentrate on the case of $p \neq 2,3$ for now.

B.2 The group law on $E(K)$

We can depict the group structure on $E(K)$ geometrically, for $K = \mathbb{R}$; we then derive formulas to realize it algebraically, and these formulas are valid over any field (of characteristic different from 2 and 3).

Namely, given any two points P and Q on the curve, draw the straight line connecting them. It either intersects the curve in a third point, or else is a vertical line. In the second case, we deem that P is the additive inverse of Q, written $P = -Q$[1]; in the first we deem the point of intersection to be $-(P+Q)$. This is consistent with identifying $\mathcal{O}$ as the additive identity of the group.

To double a point, draw a line tangent to P; this intersects the curve at the point P with multiplicity 2 and then also either intersects the curve at one other point, which we denote $-2P$, or else is vertical, in which case we say that $2P = \mathcal{O}$.

More precisely:

- If $P = \mathcal{O}$ then define $P + Q = Q + P = \mathcal{O}$ for all Q and define $-P$ to be $\mathcal{O}$.
- If $P \neq \mathcal{O}$ has affine coordinates (x,y), then define $-P$ to be the point with affine coordinates $(x,-y)$; this is the other point of intersection of the vertical line through P with E (and could be equal to P).

[1]Note that if $Q = (x,y)$ then $-Q = (x,-y)$ NOT $(-x,-y)$

- If $P, Q \neq \mathscr{O}$ and $P \neq Q$ then define $P+Q$ as $-R$, where R is the third point of intersection of the line through P and Q with E.
- If $P = Q \neq \mathscr{O}$ then define $2P = P+P$ to be $-R$, where R is the third point of intersection of the line tangent to E at P with E.

Does this define a group structure on the set $E(K)$? Firstly: *Bezout's theorem* promises that a line which intersects a curve of degree 3 in two points (counting multiplicities) must intersect it in a third point (including potentially $\mathscr{O}$); thus addition is well-defined. It is clearly commutative.

Secondly: applying the addition rule to $\mathscr{O}$ and P gives a vertical line through P; we deduce that $\mathscr{O} + P = P$.

Thirdly, each element has an additive inverse (by taking the vertical line through P).

Lastly: associativity. This requires some rather nice geometric arguments, but at least with a picture it seems plausible, for the moment.

What is an algebraic formula for addition? First: the general case. If $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ with $x_1 \neq x_2$ then the line through them is

$$y = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)(x - x_2) + y_2 = \alpha x + \beta.$$

If $P = Q$ then instead we want the tangent line, so we differentiate $y^2 = x^3 + ax + b$ to obtain $2yy' = 3x^2 + a$, whence the slope of the tangent line at (x_1, y_1) is

$$\alpha = \frac{3x_1^2 + a}{2y_1};$$

we could also compute the y-intercept β.

The third point of intersection is then a point (x, y) on this line such that $y^2 = x^3 + ax + b$, that is, corresponds to x such that

$$(\alpha x + \beta)^2 = x^3 + ax + b$$

so

$$x^3 - (\alpha x + \beta)^2 + ax + b = 0$$

This factors as

$$(x - x_1)(x - x_2)(x - x_3) = 0$$

since x_1 and x_2 are both roots already. Consequently, $x_3 = \alpha^2 - x_1 - x_2$ (from the coefficient of x^2 on both sides). We simplify to get the following formulae.

Lemma B.4 If $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ then the sum $P + Q$ is $\mathscr{O}$ if $x_1 = x_2$ and $y_1 = -y_2$ and otherwise is the point (x_3, y_3) where

$$x_3 = \alpha^2 - x_1 - x_2, \qquad y_3 = \alpha(x_1 - x_3) - y_1,$$

and

$$\alpha = \begin{cases} \frac{y_1-y_2}{x_1-x_2} & \text{if } P=(x_1,y_1) \neq Q=(x_2,y_2),\, x_1 \neq x_2; \\ \frac{3x_1^2+a}{2y_1} & \text{if } P=Q,\, y_1 \neq 0. \end{cases}$$

Using these equations, one could verify associativity directly (though very painfully).

More importantly: these equations give the group law on $E(K)$, for any field K (of characteristic different from 2 or 3), since applying the formulas will work regardless of the underlying field.

■ **Example B.5** Let $y^2 = x^3+1$ and $K = \mathbb{F}_5$. Find all multiples of $P=(2,3)$.

Solution: We first note that $(2,3)$ is indeed a K-rational point on this curve. By the above formula:

$$\alpha = \frac{3(4)+0}{2(3)} = 2; x_3 = 4-2(2) = 0, y_3 = 2(2-0)-3 = 1$$

so $2P = (0,1)$. Now $2P+P$ is given by

$$\alpha = \frac{2}{2} = 1, \quad x_3 = 1-2-0 = -1, y_3 = 1(0-(-1))-1 = 0$$

so $3P = (-1,0)$. We get $4P = 2(2P)$:

$$\alpha = \frac{3(0)+0}{2(1)} = 0; \quad x_3 = 0^2-2(0) = 0, y_3 = 0-1 = -1$$

that is, $4P = (0,-1)$. Writing $5P = 2P+3P$ gives

$$\alpha = \frac{1}{1} = 1; \quad x_3 = 1-0+1 = 2; y_3 = 1(0-2)-1 = -3$$

so $5P = (2,2)$. Finally, $6P = 2(3P) = \mathscr{O}$ since the line tangent to $3P$ has vertical slope (α is undefined).

So we have deduced that the cyclic subgroup generated by P is the set

$$\{\mathscr{O}, (2,3), (0,1), (-1,0), (0,-1), (2,2)\}$$

which is in fact the entire set $E(K)$. Hence $E(K)$ is a cyclic group of order 6. ■

This latter was a pleasant surprise: although it's clear that if a group has prime order, then it must be cyclic (and any point $P \neq \mathscr{O}$ must be a generator), in this example the group of points was cyclic even though its order was not prime.

B.3 ECC: Elliptic Curve Cryptography

There are a number of cryptosystems built upon cyclic groups which can be adapted for use with elliptic curves in a straightforward way. Let's describe two of the ones we already know; see also Koblitz for a description of the Massey-Omura cryptosystem, for example.

Diffie-Hellmann key exchange on elliptic curves For this method, Alice and Bob publicly agree on an elliptic curve E defined over a field $\mathbb{F}_q$ and a base point $P \in E(\mathbb{F}_q)$, such that the order N of P (meaning, the least positive n such that $nP = \mathcal{O}$) is very large. Then the subgroup of $E(\mathbb{F}_q)$ generated by P is the cyclic subgroup of the classical Diffie-Hellmann key exchange.

Now Alice chooses a large integer a, of the same order of magnitude at N; she calculates aP and sends it to Bob; similarly, Bob chooses b and sends bP to Alice. Finally, Alice and Bob each compute $abP = a(bP) = b(aP)$. Then they can, for example, use the x-coordinate of their answer as their shared secret key.

Meanwhile, Eve has access to P, aP and bP; if the discrete logarithm problem is hard on E, then it is assumed (the computational Diffie-Hellmann problem) that Eve cannot recover abP.

ElGamal over elliptic curves For this public key method, there is a publicly-available elliptic curve E defined over $\mathbb{F}_q$ as well as a publicly known point $P \in E(\mathbb{F}_q)$ which generates a large cyclic subgroup of $E(\mathbb{F}_q)$.

To compute his public and private keys, Bob chooses a large integer b, calculates bP and publishes this as his public key; he keeps b as his private key.

To send a message m to Bob, Alice encodes m as a point $P_m \in E(\mathbb{F}_q)$ and chooses a large random integer a. She then sends to Bob the pair:

$$(Q,R) = (aP, P_m + a(bP)).$$

Bob, upon receiving this pair, recovers P_m as

$$P_m = R - bQ = P_m + a(bP) - b(aP).$$

Again, Eve needs to discover either a or b to recover P_m, but to do so from the information available requires Eve to solve the DLP on the elliptic curve.

Computational costs As always, we should consider the cost (and possibility!) of implementing each of our cryptosystems, before exploring their security. These costs include:

- Set-up costs: choosing an elliptic curve and a large cyclic subgroup with generator P; finding the number of points on the curve.
- Operational costs: computing aP for a large; adding two points on the curve; associating messages m to points P_m on the curve.

So far, we have only answered part of the second question: complexity of point addition and scalar multiplication. Namely, to add two points involves about 10 arithmetic operations (addition, subtraction, multiplication, division) over the finite field. Thus if our field is $\mathbb{Z}_p$, the complexity is $O(10\log^2 p) = O(\log^2 p)$. If our field is an extension of $\mathbb{Z}_p$, then this complexity depends on the implementation of the field arithmetic (and so this algorithm would only be feasible if the field operations were). To calculate aP, for large a, we use the "double and add" technique achieved by expressing a in binary; as above this gives, over $\mathbb{Z}_p$, a cost of $O(\log^3 p)$.

In the sections that follow, let us take up the challenge of some of the other questions above.

The number of points on an elliptic curve It's easy to see that

$$y^2 = x^3 + ax + b$$

has at most $2q+1$ solutions in K^2: the point at infinity, plus at most q pairs $(x, \pm\sqrt{x^3+ax+b})$. If we set $\chi(x) = 1$ if x is a square in K^*, $\chi(0) = 0$, and -1 otherwise, then the number of points on $E(K)$ is

$$1 + \sum_{x\in K}(1+\chi(x^3+ax+b)) = q+1+\sum_{x\in K}\chi(x^3+ax+b).$$

Since $\chi(x)$ is about equally likely to be $+1$ as -1, one expects that $\chi(x^3+ax+b)$ would have approximately the same behaviour; the sum then becomes a random walk and probability theory suggests the sum will be bounded by $\sqrt{q}$ in absolute value, with almost uniform distribution. In fact, this is true!

Theorem B.6 — Hasse's theorem. The number of points N on the $\mathbb{F}_q$-rational points of an elliptic curve E satisfies

$$q+1-2\sqrt{q} \le N \le q+1+2\sqrt{q}$$

and furthermore, the number of curves of size $N = q+1+t$ is approximately $\frac{1}{\pi}\sqrt{4q-t^2}$.

There exist polynomial time algorithms due to Schoof, Elkies and Atkin, to calculate the precise number of points on an elliptic curve over $\mathbb{F}_q$. Unfortunately, they range in $O(\log^5 q)$ to $O(\log^8 q)$, and for a 400-bit prime, it could take a year to complete the calculation! When $q = 2^k$, there are more efficient algorithms available, that make even $k = 10,000$ bits possible.

What this implies: if you fix a prime power q and some primes near $q+1$, then there is a good probability of choosing an elliptic curve over $\mathbb{F}_q$ of prime order.

The structure of the group of points on an elliptic curve. It turns out that the group $E(\mathbb{F}_q)$ is very often a cyclic group, or close to it. For instance, if $|E(\mathbb{F}_q)| = n$ where n is a prime number, then necessarily $E(\mathbb{F}_q)$ is cyclic and any point $P \in E(\mathbb{F}_q)$, $P \neq \mathcal{O}$, is a generator. (Exercise)

More generally, we have the following theorem; the first part is due to Shoof.

Theorem B.7 Let $q = p^n$ for a prime p and suppose E is an elliptic curve such that $|E(\mathbb{F}_q)| = m = q+1-t$. Then $E(\mathbb{F}_q)$ is cyclic

- if $t = \pm\sqrt{q}$ and n is even or $\gcd(p-1,3) = 1$; OR
- if $t = \pm\sqrt{pq}$, and n is odd and $p \in \{2,3\}$; OR
- if $t = 0$, n is odd and $q \neq 3 \mod 4$; OR
- if $t = 0$, $p \neq 1 \mod 4$ and $q \neq 3 \mod 4$.

In general, $E(\mathbb{F}_q)$ is either cyclic or a product of at most two cyclic subgroups.

There exist extensive tables of elliptic curves over various finite fields, together with generators (or, generators for their maximal cyclic subgroups). Elliptic curves are of interest far beyond cryptography; they originally arose in number theory and are characterised in algebraic geometry as nonsingular projective curves of genus 1. For example, Koblitz's interests in elliptic curves (see chapter VI) extend into the Weil conjectures about zeta functions.

Finding points on an elliptic curve and associating messages to points Finding points on an elliptic curve $E(\mathbb{F})$ is fairly easy for fields $\mathbb{F} = \mathbb{F}_p$, p a prime. If E is defined by the equation $y^2 = x^3+ax+b$, for example, then

- choose $x \in \mathbb{F}_p$ at random until $x^3 + ax + b$ is a square, which can be done with the theory of quadratic residues mod p;
- compute the two square roots of $x^3 + ax + b$ mod p (for which there are efficient algorithms, such as Shanks–Tonelli).

It also reasonably efficient to find random points on elliptic curves over $\mathbb{F}_q$.

This ability to find points is important for a number of practical reasons, including the need to identify points on our elliptic curve with our plaintext messages. A standard way to do this latter (see [Kob87, Chapter VI.2]): fix k sufficiently large so that the size of the alphabet times k is less than q.

- given an integer m (eg: a binary string identified as an integer in base-2)
- write it in base p
- identify this string with a polynomial $f \in \mathbb{F}_p[X]$
- (assuming $q = p^r$ and $\deg(f) < r$) identify f with an element x of $\mathbb{F}_q$
- test if $x^3 + ax + b$ has a square root y; if so, identify m with (one choice of) $(x, y) \in E(\mathbb{F}_q)$.
- otherwise, replace x with each of $x+1, x+2, \cdots$ in turn, until a point is found.

This algorithm is deterministic and invertible: if a point $(x, y) \in E(\mathbb{F}_q)$ represents a value m, then $\lfloor x/k \rfloor = m$.

There is lots more to say, including about generating curves and the points on them. Moreover, we haven't explained the “magic" of why $E(K)$ is a group — it's a consequence of a fundamental concept of algebraic geometry (and number theory!), which defines the group of divisors on a curve.

C. Solutions

C.1 Section 2.4

Question 1 : The standard set of representatives is $\{0,1,2,3,4\}$ but any set will do; another common choice (see Section 11) is $\{-2,-1,0,1,2\}$. The following tables come out the same for this second case when we replace 3 with $-2 \equiv 3$ and 4 with $-1 \equiv 4$:

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Note that the addition table is very regular but the multiplication table is more chaotic; this has cryptographic implications. Note that in this case, each row of the multiplication table contains every element exactly once. Why?

Question 2 : We verify that $(\mathbb{Z}_6,+)$ satisfies the axioms of an additive group, where $\mathbb{Z}_6 = \{0,1,2,3,4,5\}$:
closure Adding two integers and taking the remainder mod 6 gives an integer between 0 and 5.
commutativity Addition in the integers is commutative, so this also holds here.
associativity Addition in the integers is associative, so this also holds here.
identity The identity element 0 also acts as the identity element in the integers, so this also holds here.
inverse The inverse of $a \in \{0,1,2,3,4,5\}$ is $b = 6-a$ because b is in the set and $a+b = a+(6-a) = 6 \equiv 0$.
Note that this was NOT the subgroup test, because the operation is NOT the same as in the integers. In fact, as a subset of the integers, $\{0,1,2,3,4,5\}$ is not a subgroup — it's not closed under normal addition. Nonetheless, we can leverage our understanding of the integers to infer this is a group (as

above).

We can write out the multiplication table for $\mathbb{Z}_6$ to find all the elements without multiplicative inverses:

$\cdot$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Thus $0,2,3,4$ do not have multiplicative inverses modulo 6. Alternatively, we may note that these are all the elements whose gcd with 6 is not 1.

It follows that $\mathbb{Z}_6$ is not a group under multiplication, since the axiom about inverses fails.

Question 4 : We can write out the addition and multiplication tables (noting for example that $x+x=2x=0$):

$+$	0	1	x	$1+x$
0	0	1	x	$1+x$
1	1	0	$1+x$	x
x	x	$1+x$	0	1
$1+x$	$1+x$	x	1	0

$\cdot$	1	x	$1+x$
1	1	x	$1+x$
x	x	$1+x$	1
$1+x$	$1+x$	1	x

We have used calculations like $(1+x)(1+x)=1+2x+x^2=1+0+(1+x)=2+x=x$ to complete the second table. Note that we removed 0 since the requirement is that $F\setminus\{0\}$ is a group.

We already knew the addition and multiplication are commutative, associative and distributive because this holds for polynomials; these tables show in addition that the sets are closed under the operations, that there is an identity in each case, and that every element has an inverse. Thus it's a field with 4 elements, denoted $\mathbb{F}_4$.

To show that $\mathbb{F}_4 \not\cong \mathbb{Z}_4$, note that $\mathbb{Z}_4\setminus\{0\}$ is not a field since 2 is not invertible. Or else note that $(\mathbb{Z}_4,+)$ is cyclic, with a generator of order 4, but that $(\mathbb{F}_4,+)$ is not cyclic: every nonzero element has order 2.

Question 8 : We can freely choose y and z in $\mathbb{Z}_7$; for any such pair (y,z), we have a unique choice $x=-y-z\in\mathbb{Z}_7$ for which $(x,y,z)\in W$. Thus if we choose $(y,z)=(1,0)$ we get $(6,1,0)\in W$ and if we choose $(y,z)=(0,1)$ we get $(6,0,1)\in W$.

These are linearly independent (because a set of two vectors is linearly independent iff neither is a scalar multiple of the other), so $\dim(W)\geq 2$. Since $W\subsetneq\mathbb{Z}_7^3$, which has dimension 3, we have $\dim(W)=2$ and $\{610,601\}$ is a basis.

Question 9 : If W is a vector space of dimension n over a field with q elements, then there is a basis $B=\{w_1,\cdots,w_n\}$ of W. This means that every vector in W can be written as a unique linear combination of the vectors in B. That is, W is in bijection with F^n by the map sending

$$w = \sum_{i=1}^{n} c_i w_i \in W$$

to its coordinates $(c_1, \cdots, c_n) \in F^n$. Thus $|W| = |F^n| = |F|^n = q^n$.

Bibliography

[ABD$^+$21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals-kyber (version 3.02) – submission to round 3 of the nist post-quantum project: Specification document. *https://pq-crystals.org/, under Resources*, 2021.

[ABD$^+$22] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals: Cryptographic suite for algebraic lattices. *https://pq-crystals.org/*, 2022.

[Ber68] Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill Book Co., New York-Toronto-London, 1968.

[Ber70] E. R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24:713–735, 1970.

[BRC60] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3:68–79, 1960.

[CZ81] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Math. Comp.*, 36(154):587–592, 1981.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22(6):644–654, 1976.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31(4):469–472, 1985.

[Gal30] Évariste Galois. Sur la théorie des nombres. *Bulletin des sciences mathématiques, physiques et chimiques*, 13:428–435, 1830.

[GJN21] Thierry Giordano, Barry Jessup, and Monica Nevins. *Vector Spaces First: An Introduction to Linear Algebra*. uO Research, http://hdl.handle.net/10393/43955, 2021.

[Hoc59] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–156, 1959.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: a ring-based public key cryptosystem. In *Algorithmic number theory (Portland, OR, 1998)*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 267–288. Springer, Berlin, 1998.

[HPS14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An introduction to mathematical cryptography*. Undergraduate Texts in Mathematics. Springer, New York, second edition, 2014.

[Jor70] Camille Jordan. *Traité des substitutions et des équations algébriques*. Les Grands Classiques Gauthier-Villars. [Gauthier-Villars Great Classics]. Éditions Jacques Gabay, Sceaux, 1989, 1870. Reprint of the 1870 original.

[Kob87] Neal Koblitz. *A course in number theory and cryptography*, volume 114 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1987.

[LIN75] J. H. VAN LINT. A survey of perfect codes. *The Rocky Mountain Journal of Mathematics*, 5(2):199–224, 1975.

[Mac03] David J. C. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, New York, 2003.

[McE78] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *JPL DSN Progress Report*, pages 114–116, 1978.

[MZ23] Robert Morelos-Zaragoza. The error correcting codes (ecc) page. *http://www.eccpage.com/*, Last accessed 2023.

[Nic12] W. Keith Nicholson. *Introduction to abstract algebra*. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, fourth edition, 2012.

[OR01] J.J. O'Connor and E.F. Robertson. Alexandre-théophile vandermonde. *MacTutor History of Mathematics, (University of St Andrews, Scotland)*, 2001.

[Ple98] Vera Pless. *Introduction to the theory of error-correcting codes*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York, third edition, 1998. A Wiley-Interscience Publication.

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8:S5–S9, 1962.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), sep 2009.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell System Tech. J.*, 28:656–715, 1949.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

Index