

# MATH

This supplement was written solely by me<sub>3</sub> (Mike Speciner) with the intention of providing material to expand and deepen your knowledge of the mathematics and algorithms behind many of the techniques explained in the book. You should be able to understand the rest of the book without this knowledge, but I<sub>3</sub> think it could still be helpful, or at least enlightening. This document, including the homework problems: <https://github.com/ms0/docs/blob/main/math.pdf>.

## M.1 INTRODUCTION

Computing technology has encouraged us to turn pretty much everything into bits. We use bits to represent numbers, text, images, audio, video, objects, scents, and even more abstract concepts like spacetime, quantum fields, and money. For a few of these, the representation can be exact, but mostly the representation is merely a “good enough” approximation. In all cases, the representation is of necessity finite—we exist in a universe of limited resources.

Once we have a representation in bits, we can make exact copies, send those copies around, mangle them in ways so that the original representation can be retrieved with a key (encryption), hide them amongst a sea of bits representating something else (steganography), derive a “small” sequence of bits that is (practically) unique to that representation (hashing), and guarantee their integrity with MACs and provenance with signatures.

The building blocks of these representations and of the cryptographic techniques for manipulating them are based on mathematical concepts that we will elucidate in this document. We assume a basic knowledge of elementary school arithmetic and high school algebra. It’s a good idea to take a look at the homework problems when they’re referenced. Actually doing them should help to promote understanding.

We’ll discuss such mathematical structures as groups, rings, fields, polynomials, and matrices. Groups are central to RSA and Diffie-Hellman; fields are used in other traditional public key schemes. Various rings (polynomials, matrices, mod  $n$  integers) are used in many of the post-quantum schemes. Fields are found in AES and secret sharing, as well as in some of the modes of operation. And matrices provide a clean description of secret sharing, quantum computing operations, and discrete Fourier transforms (DFTs) that are the basis of Shor’s algorithm.

## M.2 SOME DEFINITIONS AND NOTATION

A set is a collection of elements. For example, the empty set (the set with no elements) is  $\{\}$ . The set of bits is  $\{0,1\}$ . The set of all elements with property  $P$  is  $\{x \mid x \text{ has property } P\}$ . We write  $e \in S$  to mean that  $e$  is an element of set  $S$ , and  $x \notin S$  to mean  $x$  is not an element of  $S$ . For sets  $S$  and  $T$ , the difference  $S - T = \{s \mid s \in S \text{ and } s \notin T\}$ . More generally,  $\{expression \mid condition\}$  is the set of elements that can be written as *expression* subject to *condition*, e.g.,  $\{x^2 + y^2 \mid x \in \mathbf{Z} \text{ and } y \in \mathbf{Z}\}$  is the set of numbers that can be expressed as the sum of two squares of integers.

As is common in elementary arithmetic, we use the symbol  $+$  for addition,  $\times$  for multiplication,  $-$  for subtraction, and  $/$  for division. These are called binary operators, because they take two inputs. As is common, we'll also use  $-$  as the unary operator for negation. We may also use  $\cdot$  for multiplication, as in  $a \cdot b$ , or leave it out altogether, as in  $ab$ . We use parentheses to specify the order of operations. Without parentheses, we'll assume that unary operators are applied right-to-left then multiplication and division are applied left-to-right, and then addition and subtraction are applied left-to-right. But if we indicate any multiplications without an explicit symbol (i.e., by concatenation rather than  $\times$  or  $\cdot$ ), those multiplications occur before any divisions, e.g.,  $ab/cd = (ab)/(cd)$ . Eventually, we'll also be showing exponentiation in the usual way, e.g.,  $2^3 = 2 \times 2 \times 2 = 8$ . The first use of that notation will be  $a^{-1}$  to indicate the multiplicative inverse of  $a$ :  $a^{-1} \times a = 1$ . This notation is consistent with the property  $a^{x+y} = a^x \times a^y$ , and the consequence that  $a^0 = 1$ . By convention,  $a^{b^c}$  means  $a^{(b^c)}$ , since  $(a^b)^c = a^{bc}$ . Exponentiation is performed after the evaluation of the exponent but before any other operator.

We'll write "iff" to mean "if and only if".

Eventually, we'll use  $\sum$  notation for sums and  $\prod$  notation for products:

$$\sum_{i=m}^n a_i = a_m + a_{m+1} + \cdots + a_{n-1} + a_n \qquad \prod_{i=m}^n a_i = a_m a_{m+1} \cdots a_{n-1} a_n$$

As an example, we can define exponentiation (for non-negative integer exponents) in  $\prod$  notation as

$$a^n = \prod_{i=1}^n a$$

(A sum with no terms is 0; a product with no terms is 1.) Sometimes, instead of low and high limits, we'll specify the components of a sum or product with a condition, e.g.,  $a^n = \prod_{1 \leq i \leq n} a$ . When not otherwise specified, the index variable ( $i$  in the previous examples) takes on integer values satisfying the condition or limits. Within this document, if a low limit is not specified, it is assumed to be 0. Sometimes we'll leave out the index variable altogether when it's clear what it is.

## M.3 ARITHMETIC

We start with simple arithmetic. You are undoubtedly at least somewhat familiar with the nonnegative integers  $\mathbf{N} = \{0, 1, 2, \dots\}$  and the binary operations of addition and multiplication. While  $\mathbf{N}$  is not finite, its structure has a plethora of pleasant properties:

(A) **associativity**, for both addition and multiplication:

$$(a+b)+c = a+(b+c) \text{ and } (a \times b) \times c = a \times (b \times c) \quad \text{for all } a, b, c$$

When a binary operation is associative, we can dispense with the parentheses.

(C) **commutativity**, for both addition and multiplication:

$$a+b = b+a \text{ and } a \times b = b \times a \quad \text{for all } a, b$$

(D) **distributivity** of multiplication over addition:

$$a \times (b+c) = (a \times b) + (a \times c) \text{ and } (a+b) \times c = (a \times c) + (b \times c) \quad \text{for all } a, b, c$$

By convention, multiplication takes precedence over addition, so, for example, the parentheses on the right-hand side of the above equations (but not those on the left-hand side) can be elided and will be assumed.

(I) existence of an **identity**, for both addition (0) and multiplication (1):

$$0+a = a = a+0 \quad \text{and} \quad 1 \times a = a = a \times 1 \quad \text{for all } a$$

One property that  $\langle \mathbf{N}, +, \times \rangle$  does not have is

(N) existence of **inverses**, for both addition and multiplication:

$$a + -a = 0 = -a + a \quad \text{for all } a$$

$$a \times a^{-1} = 1 = a^{-1} \times a \quad \text{for all nonzero } a$$

Property N would allow us to do subtraction and division, the “opposites” of addition and multiplication:  $a - b = a + -b$ ;  $a/b = a \times b^{-1}$ .

We can supplement  $\mathbf{N}$  with negative integers  $\{-1, -2, \dots\}$  to get all the integers (designated as  $\mathbf{Z}$ , for *Zahlen*, the German word for number). That gives us property N for addition, so we can do subtraction. To get property N for multiplication, we have to supplement  $\mathbf{Z}$  with multiplicative inverses  $a^{-1}$  (usually written  $1/a$ ) for each nonzero  $a$  (other than 1, which is its own inverse). But we have to be able to add with the resulting set of numbers, which leads us to require all rational numbers (designated as  $\mathbf{Q}$  for *quotient*), i.e., numbers of the form  $n \times d^{-1}$  (usually written  $n/d$ ) where numerator  $n$  and denominator  $d$  are integers and  $d \neq 0$ . Addition and multiplication rules for rational numbers can be derived from properties A, C, D, I, and N (see Homework Problem 2).

---

## M.4 ABSTRACT ALGEBRA

The mathematic structures we just described give us a few examples of the kinds of structures useful for cryptography:

A **group**  $\langle G, \circ \rangle$  is a structure that satisfies properties A, I, and N for the binary operation  $\circ$ . The binary operation is often written as  $\cdot$ , or even elided, and its identity element is sometimes written as 1, but often written as  $e$  (for *eins*, the German word for one). A **commutative group**, aka an **Abelian group** (named after the Scandinavian mathematician Niels Henrik Abel, who originated group theory and died at 26), also satisfies property C. For Abelian groups, the binary operation is often written as  $+$  with identity element 0.  $\langle \mathbf{Z}, + \rangle$ ,  $\langle \mathbf{Q}, + \rangle$ , and  $\langle \mathbf{Q} - \{0\}, \times \rangle$  are Abelian groups.

There is a special type of group called a **cyclic group**. Its elements are all the powers (assuming the group operation is multiplication) of a single element called a generator. Property A is enough to prove that cyclic groups are Abelian. The **discrete logarithm problem** is the problem of finding the power of a given generator that produces a given group element. Various cryptographic schemes (*e.g.*, Diffie-Hellman) depend for their security on the difficulty of their discrete logarithm problem.  $\langle \mathbf{Z}, + \rangle$  is a cyclic group; it has two generators: 1 and  $-1$ ; its discrete logarithm problem is absurdly easy.

A **ring**  $\langle R, +, \times \rangle$  is a structure that satisfies properties A and D and properties C, I, and N for addition. A **ring with identity** also satisfies property I for multiplication. A **commutative ring** also satisfies property C for multiplication. The integers form a commutative ring with identity:  $\langle \mathbf{Z}, +, \times \rangle$ .

A **division ring** is a ring with identity that satisfies property N for multiplication (excluding zero). This makes  $\langle R - \{0\}, \times \rangle$  a group. If that group is not commutative, the division ring is called a **skew field**. The quaternions (a four-dimensional extension of the real numbers that combines scalars and three-dimensional vectors, where multiplication combines scalar-, dot-, and cross-products) form a skew field. As it turns out, there are no finite non-commutative division rings.

A **field** is a commutative division ring. The rational numbers form a field:  $\langle \mathbf{Q}, +, \times \rangle$ . So do the real numbers  $\langle \mathbf{R}, +, \times \rangle$  and the complex numbers  $\langle \mathbf{C}, +, \times \rangle$ .

The problem with all the groups, rings, and fields we've mentioned so far is that they're infinite, which makes them computer-unfriendly. What we really want are finite groups, rings, and fields, so that we can store each element of the structure in a fixed number of bits while many of the usual rules of algebra still apply. An additional advantage of a finite structure is that it is possible to choose an element of the structure completely at random, in the sense that each element is equally likely to be chosen.

## M.5 MODULAR ARITHMETIC

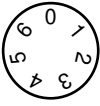
Some of the simplest finite structures are based on modular arithmetic, so we'll present those first.

Returning to  $\langle \mathbf{Z}, +, \times \rangle$ , the ring of integers, there is a notion of division with remainder. For any nonzero  $d$  in  $\mathbf{Z}$ , we consider the set  $d\mathbf{Z} = \{dk \mid k \in \mathbf{Z}\}$ , the multiples of  $d$ . We say that  $d$  is a **divisor** of  $n$  or that  $d$  **divides**  $n$  (written  $d|n$ ) iff  $n$  is in  $d\mathbf{Z}$ . The **remainder** when dividend  $n$  is divided by divisor  $d$  (written  $n \bmod d$ ) is the smallest nonnegative  $r$  for which there is a  $q$  in  $\mathbf{Z}$  such that  $n = q \times d + r$ ; that  $q$  is then the **quotient**. Clearly,  $0 \leq (n \bmod d) < d$ . Note that  $d|n$  iff  $n \bmod d = 0$ . (A note of warning: Computer implementations of division sometimes produce negative remainders when the dividend or divisor is negative. For example, in Python, a nonzero remainder always has the same sign as the divisor.)

We define a **prime number** to be an element of  $\mathbf{N}$  that has exactly two positive divisors, namely itself and 1. The first few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31. We define two integers to be **relatively prime** (*aka coprime*) iff the only positive divisor they have in common is 1. We'll shortly see that the greatest common divisor of integers  $a$  and  $b$  (written  $\gcd(a, b)$ ) can be expressed as  $ua + vb$  for some integers  $u$  and  $v$ .

Remainders are the key to a different kind of arithmetic—modular arithmetic (*aka clock arithmetic*). We choose a positive integer  $m > 1$ , called the modulus. Addition and multiplication are performed using normal addition and multiplication, but whenever we get a result less than 0 or greater than  $m-1$ , we replace it by its remainder when divided by  $m$ . In other words,  $a+b$  is replaced by  $(a+b) \bmod m$ , and  $a \times b$  is replaced by  $(a \times b) \bmod m$ . It's sometimes called clock arithmetic because we imagine  $m$  bins arranged uniformly around a circle (like the hour markings on an analog clock) labeled with 0 at the top, then clockwise 1, 2, ...,  $m-1$ . We imagine placing 0, 1, 2, ...,  $m-1$  in the corresponding bins and continue clockwise placing  $m, m+1, m+2, \dots$  in bins 0, 1, 2, ..., and placing  $-1, -2, \dots$  counterclockwise in bins starting at  $m-1$ . As a result,  $a \bmod m$  is the label of the bin where  $a$  is placed. If we do normal addition, subtraction, or multiplication of any pair of integers, the bin where our result is found will depend only on those integers' bins. As a bonus, if  $m$  is prime, we even get multiplicative inverses and so can do division without needing fractions! If  $m$  isn't prime, only some of the integers mod  $m$  have inverses. (We'll explore this further in §M.5.2 *Computing Inverses in Modular Arithmetic*.) For example, Figure M-1 shows mod 7 addition and

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5



×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Figure M-1. mod 7 Arithmetic

---

multiplication tables. The advantage of modular arithmetic is that there are only finitely many elements to worry about, so a computer can easily handle it exactly, at least when  $m$  isn't too large.

### M.5.1 How Do Computers Do Arithmetic?

Most computers natively represent nonnegative integers as fixed-length sequences of bits using the binary number system, where each successive bit has a value twice that of its predecessor (as opposed to the familiar decimal number system where each successive digit (from right to left) has a value ten times that of its predecessor). Most computers natively “know” how to do addition, subtraction, multiplication, and division using this representation. But there are a number of wrinkles. First, because these integers are fixed-length (usually a choice of 8, 16, 32, 64, or 128 bits), addition and subtraction are done mod  $2^b$ , where  $b$  is that fixed number of bits. Multiplication often produces a double-length result. Division produces a quotient and a remainder. And signed integers are often represented in **two's complement**, meaning that if the high order bit of  $b$ -bit integer  $n$  is 1, it can be interpreted as the negative integer  $n - 2^b$ , and the same (unsigned) addition and subtraction operations still work, as does single-length multiplication, but division and double-length multiplication are different for signed operands.

Some high-level languages like Python use multiprecision software internally so that arithmetic operations work on variable-length integers that are automatically sized to produce ordinary integer arithmetic results (as long as there is enough memory to store them).

It's worth mentioning that many computers do (approximate) arithmetic with real numbers using a scheme called **floating point**. Floating point numbers are typically represented with a 1-bit sign, an  $s$ -bit significand, and an  $x$ -bit exponent.  $x+s+1$  is typically 32 or 64. In the IEEE 754 binary double precision floating point standard,  $x = 11$  and  $s = 52$ . The value of such a floating point number is the significand interpreted as a binary fraction in the half-open interval  $[\frac{1}{2}, 1)$  times  $2^{\text{exponent}}$ , where the exponent is interpreted as a signed integer. Special bit configurations are reserved to represent  $\pm 0$ , and  $\pm \text{infinity}$  as well as “NaN”s (“Not a Number”) and denormalized numbers, where the significand is interpreted as a binary fraction in the open interval  $(0, 1)$ . Standard floating point operations include addition, subtraction, multiplication, and division. While floating point addition and multiplication are commutative, they are neither associative nor distributive, although they are often treated as if they are. While treating floating point numbers as if they are real numbers is good enough for many purposes, cryptography is in general not one of them.

### M.5.2 Computing Inverses in Modular Arithmetic

Finding the (multiplicative) inverse of an integer  $a$  mod  $m$  means finding an integer  $u$  such that  $au \bmod m = 1$ , or, equivalently, integers  $u$  and  $v$  such that  $au + mv = 1$ . Clearly,  $au + mv$  is a multiple

of the greatest common divisor of  $a$  and  $m$ , so it can only be 1 if  $\gcd(a, m) = 1$ . In that case, here's how to do it:

If we have two positive integers,  $a$  and  $b$ , there is a marvelous algorithm for efficiently finding their greatest common divisor. It is based on the observation that if we divide  $a$  by  $b$  to get quotient  $q$  and remainder  $r$  (i.e.,  $a = bq + r$ ), then  $g$  divides both  $a$  and  $b$  iff  $g$  divides both  $b$  and  $r$ , and moreover if  $r$  is zero, then  $b$  is the greatest common divisor of  $a$  and  $b$ .

### M.5.2.1 The Euclidean Algorithm

```
def gcd(a,b) :
    while b != 0 :
        a,b = b,a%b    # replace <a,b> with a smaller pair with the same gcd
    return a
```

It turns out that we can extend the Euclidean algorithm to do more, namely to also produce integers  $u$  and  $v$  such that  $\gcd(a, b) = ua + bv$ :

```
def xgcd(a,b) :    # Let A,B be the initial (input) values of a,b
    u,v,w,x = 1,0,0,1    # then a is u*A + v*B and b is w*A + x*B throughout
    while b != 0 :
        q,r = divmod(a,b) # quotient and remainder
        a,u,v,b,w,x = b,w,x,r,u-q*w,v-q*x
    return a,u,v
```

So, if we take the set of integers between 0 and  $m$  that are relatively prime to  $m$ , i.e., whose gcd with  $m$  is 1, which we will write as  $\mathbf{Z}_m^\times$ , we can multiply and divide them mod  $m$ . In fact, they form a group under multiplication. For any prime  $p$ , all the integers between 0 and  $p$  are relatively prime to  $p$ , and so we can add, subtract, multiply, and divide mod  $p$ . The set of integers mod  $p$ , which we may write as  $\mathbf{Z}_p$ , thus forms a finite field, written as  $\text{GF}(p)$ . GF stands for *Galois field*, named after French mathematician Évariste Galois, who originated two major branches of abstract algebra, then died before he turned 21 from wounds suffered in a duel. We'll talk more about finite fields in §M.7.2.

We can ask how many nonnegative integers less than  $m$  are relatively prime to  $m$ , a value known as  $\phi(m)$ .  $\phi$  is called the **Euler totient function**. For any prime  $p$ , it's not hard to see that  $\phi(p^n) = p^n - p^{n-1}$  as there are  $p^{n-1}$  multiples of  $p$  less than  $p^n$ . Now if  $a$  and  $b$  are relatively prime,  $\phi(ab) = \phi(a)\phi(b)$ . The easiest way to see this is by use of the Chinese remainder theorem (see next section): If  $x < a$  is relatively prime to  $a$  and  $y < b$  is relatively prime to  $b$ , there is a unique  $z < ab$  such that  $z \bmod a = x$  and  $z \bmod b = y$ , and that  $z$  is relatively prime to  $ab$ . (See Homework Problem 4.)

We'll see later that if  $x \in \mathbf{Z}_m^\times$ ,  $x^{\phi(m)} = 1$ . But we can ask what is the smallest exponent  $\lambda(m)$  such that  $x^{\lambda(m)} = 1$  for all  $x \in \mathbf{Z}_m^\times$ .  $\lambda$  is called the **reduced totient function**, and is what is used in RSA. For odd primes  $p$ ,  $\lambda(p^n) = \phi(p^n)$ , and if  $a$  and  $b$  are relatively prime,  $\lambda(ab) = \text{lcm}(\lambda(a), \lambda(b))$ ,

---

where  $\text{lcm}(a,b)$  is the least common multiple of  $a$  and  $b$  and is equal to  $ab/\text{gcd}(a,b)$ . (See Homework Problem 13.)

### M.5.2.2 The Chinese Remainder Theorem

A useful observation is that, if  $j$  and  $k$  are relatively prime and you know  $a \bmod j$  and  $a \bmod k$ , you can compute  $a \bmod jk$ , and vice versa. It's easy to see that  $a \bmod j = (a \bmod jk) \bmod j$  and  $a \bmod k = (a \bmod jk) \bmod k$ . The other direction is a bit trickier. From the extended Euclidean algorithm, we know that  $uj + vk = 1$ . Let  $a_j = a \bmod j$ , and  $a_k = a \bmod k$ . If we now let  $a = (a_j vk + a_k uj) \bmod jk$ , then  $a \bmod k = a_k uj \bmod k = a_k (uj + vk) \bmod k = a_k \times 1 \bmod k = a_k$ , and  $a \bmod j = a_j vk \bmod j = a_j (uj + vk) \bmod j = a_j \times 1 \bmod j = a_j$ .

Why is this useful? Besides being helpful in proving properties of integers, it allows an expensive calculation on full-size numbers to instead be done twice but with half-size numbers, and then the half-size results can be combined to get the full-size result less expensively than directly by the full-size calculation. (For example, see Homework Problem 5.)

## M.5.3 How Fast Can We Do Arithmetic?

The algorithms most of us learned in elementary school for doing arithmetic are very simple, but they're nowhere near the fastest way to do multiplication and division—they take time proportional to  $n^2$  to multiply or divide  $n$ -digit numbers. The best known algorithms take time proportional to  $n \log n$ . While we won't describe any of those algorithms here, we will mention a method for doing exponentiation to power  $n$  that does roughly  $\log n$  multiplications rather than  $n$  multiplications. We write  $n$  in binary:  $n = 2^k + n_{k-1}2^{k-1} + n_{k-2}2^{k-2} + \dots + n_0 = (((((2 + n_{k-1}) \cdot 2 + n_{k-2}) \cdot 2 + \dots) \cdot 2 + n_1) \cdot 2 + n_0)$ , so  $x^n = (((((x^2 \cdot x^{n_{k-1}})^2 \cdot x^{n_{k-2}})^2 \dots)^2 \cdot x^{n_1})^2 \cdot x^{n_0})$ . Note that each bit  $n_i$  of  $n$  is either 0 or 1, so multiplying by  $x^{n_i}$  is either doing nothing or multiplying by  $x$ , which means we do  $\log n$  squarings and up to  $\log n$  multiplications by  $x$ . This works for any kind of  $x$  that can be multiplied. In Python:

```
def pow(x,n) :
    if n == 0 : return 1
    z = 1 << n.bit_length() >> 2
    y = x                # initialize the result-so-far for the leftmost exponent bit
    while z :             # while there are still exponent bits to process
        y *= y            # square the result-so-far
        if n&z : y *= x    # if the exponent bit is 1, multiply result-so-far by x
        z >>= 1           # move on to the next exponent bit
    return y
```

Note that, unmodified, this algorithm is subject to a timing attack to reveal the exponent.



## M.6 GROUPS

Remember, a **group** is a structure  $\langle G, \circ \rangle$  comprising a set of elements ( $G$ ), and a binary operation ( $\circ$ ) taking two elements of  $G$  and producing a single element of  $G$ , that satisfies properties A, I, and N. We'll usually use  $e$  to denote the identity element. When there is ambiguity, we'll specify the group operator explicitly. Normally, we'll use the same terminology and notation for the group operator as we use for multiplication, including exponentiation for repeated multiplication, where  $a^0 = e$ ,  $a^{-1}$  is the inverse of  $a$ , and  $a^{-n} = (a^{-1})^n$ .

Some examples of groups:

- $\langle \mathbf{Z}_m^\times, \times \rangle$  (multiplication mod  $m$ )
- $\langle \mathbf{Z}_m, + \rangle$  (addition mod  $m$ )
- $\langle \mathbf{Z}, + \rangle$  (This is the simplest infinite group.)
- Permutations (*i.e.*, rearrangements) of three objects, with the composition operator. (This is the simplest non-Abelian group.) See Homework Problem 8.

A **subgroup** of a group  $G$  is a subset of  $G$  that is a group under  $G$ 's operator. It is always the case that the identity element is a subgroup of any group, as is the group itself. It is also the case that the powers  $\{g^n \mid n \in \mathbf{Z}\}$  of any element  $g$  of the group form a subgroup, called the **cyclic subgroup** generated by  $g$ . If  $G$  is finite, then we only need to include non-negative powers of  $g$ . A group  $G$  is **cyclic** if it is its own cyclic subgroup, *i.e.*, there is some  $g \in G$  such that  $G = \{g^n \mid n \in \mathbf{Z}\}$ ;  $g$  is called a **generator** of  $G$ . A cyclic group can have many generators. Some examples of subgroups:

- $\{1, 9\}$  is a cyclic subgroup of  $\mathbf{Z}_{10}^\times$ ; 9 is its only generator.
- The even numbers form a cyclic subgroup of  $\mathbf{Z}$  with addition. 2 is a generator.
- Permutations of the first three of four objects is a subgroup of the four-object permutations.

The **order** of a group is the number of elements in the group. We write the order of  $G$  as  $|G|$ . If  $G$  is a finite group, and  $H$  is a subgroup of  $G$ , then  $|H|$  divides  $|G|$ . The proof involves creating a multiplication table, where the top row comprises the elements of  $H$ , starting with the identity. Successive rows consist of elements of the form  $gh$ , where  $g$  is any element of  $G$  not in any previous row of the table, and  $h$  is in  $H$ .

List the elements of subgroup $H$ of group $G$ :	$e$	$h_1$	$h_2$	$h_3$	$h_4$	$\cdots$	$h_n$
Pick $g_1 \in G$ not already listed above:	$g_1$	$g_1 h_1$	$g_1 h_2$	$g_1 h_3$	$g_1 h_4$	$\cdots$	$g_1 h_n$
Pick $g_2 \in G$ not already listed above:	$g_2$	$g_2 h_1$	$g_2 h_2$	$g_2 h_3$	$g_2 h_4$	$\cdots$	$g_2 h_n$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Pick $g_m \in G$ not already listed above:	$g_m$	$g_m h_1$	$g_m h_2$	$g_m h_3$	$g_m h_4$	$\cdots$	$g_m h_n$
(until all elements of $G$ are listed)							

No two elements in the same row can be equal, because if  $gh_i = gh_j$ ,  $h_i = g^{-1}gh_i = g^{-1}gh_j = h_j$ . (Notice how we just used the group properties of  $G$ .) It is easy to see that if  $g$  is not already in the

table, neither are any of the  $gh$ : if  $gh$  is in the table, it is  $g_i h_j$  for some  $g_i$  in the first column and some  $h_j$  in  $H$ ; then  $gh = g_i h_j$  so  $g = g_i h_j h^{-1}$ , so  $g$  is in the  $g_i$  row. (Notice how we just used the group properties of  $H$ .) So all we have to do is fill the table row by row, each time choosing a new  $g \in G$  for the first column, *i.e.*, an element of  $G$  not already in the table. When there is no such remaining  $g$ , we have completed a table where each element of  $G$  appears exactly once, and each row contains exactly  $|H|$  elements. The number of rows is called the **index** of  $H$  in  $G$ . So  $|G|$  is the product of  $|H|$  and the index of  $H$  in  $G$ .

The **order** of an element is the order of the cyclic subgroup it generates. If  $g$  has finite order, it is clear that the order of  $g$  is the smallest positive integer  $\lambda$  for which  $g^\lambda = e$ . Note that if  $g^k = e$ , then  $\lambda$  divides  $k$ : if  $k = q\lambda + r$  with  $0 \leq r < \lambda$ ,  $g^r = g^{-q\lambda} g^k = e$ , so  $r = 0$ .

***From here on, until further notice, all the groups will be commutative.***

Pay attention to how we use this fact in the next paragraph (and see Homework Problem 9).

If we have two elements with finite order, we can find an element whose order is the least common multiple of those orders: First, suppose  $a$  and  $b$  have orders  $\lambda$  and  $\mu$ , respectively, and  $\gcd(\lambda, \mu) = 1$ . Since  $\gcd(\lambda, \mu) = 1$ , there are integers  $u$  and  $v$  such that  $u\lambda + v\mu = 1$ . Consider the cyclic group generated by  $ab$ .  $(ab)^{u\lambda} = a^{u\lambda} b^{u\lambda} = e b^{u\lambda} = e b^{u\lambda} e = e b^{u\lambda} b^{v\mu} = b^{u\lambda + v\mu} = b$ , so  $b$  is in the cyclic group. Similarly,  $(ab)^{v\mu} = a$ , so  $a$  is in the cyclic group. Since the order of each element of a group divides the order of the group, the order of the cyclic group (which is the order of  $ab$ ) must be a multiple of  $\lambda\mu$ . And  $(ab)^{\lambda\mu} = a^{\lambda\mu} b^{\lambda\mu} = ee = e$ , so  $ab$  has order  $\lambda\mu$ . Now if  $a$  and  $b$  have orders  $\lambda$  and  $\mu$ , respectively, and  $\gcd(\lambda, \mu) = \gamma$ , let  $\delta = \gcd(\gamma, \mu/\gamma)$  and  $\varepsilon = \gamma/\delta$ . Consider  $a^\delta$  and  $b^\varepsilon$ . These have orders  $\lambda/\delta$  and  $\mu/\varepsilon = \mu\delta/\gamma$ , respectively, and  $\gcd(\lambda/\delta, \mu\delta/\gamma) = 1$ , so the order of  $a^\delta b^\varepsilon$  is  $\lambda\mu/\gamma = \text{lcm}(\lambda, \mu)$ .

A corollary of this result is that for any finite Abelian group, there is an element whose order is the least common multiple of the orders of all the elements. We can produce such an element as follows: set our candidate to the identity, then repeatedly choose an element whose order is not a divisor of the order of the current candidate and replace the candidate with an element whose order is the lcm of the orders of the candidate and the newly chosen element; eventually we'll run out of elements and the candidate will have the desired order. We'll make use of this corollary in §M.7.2 *Finite Fields*.

## M.7 FIELDS

Remember, a **field** is a structure comprising a set  $F$  and two operators,  $+$  and  $\times$ , satisfying property D and for which  $\langle F, + \rangle$  is a commutative group with identity 0 and  $\langle F - \{0\}, \times \rangle$  is a commutative group with identity 1. Examples of fields are

- $\langle \mathbf{Z}_p, +, \times \rangle$ , the integers mod  $p$ , where  $p$  is a prime (see Homework Problem 12)
- $\langle \mathbf{Q}, +, \times \rangle$ , the rational numbers
- $\langle \mathbf{R}, +, \times \rangle$ , the real numbers
- $\langle \mathbf{C}, +, \times \rangle$ , the complex numbers

Intuitively, you can think of a field as something like  $\mathbf{Q}$ , but without any notion of order. We'll use a lot of the same notation as we use for  $\mathbf{Q}$ , including unary minus ( $-$ ) for additive inverse ( $a + -a = 0$ ),  $a - b$  for  $a + -b$ , and  $a/b$  for  $a \times b^{-1}$ .

For cryptography, it turns out that the most useful fields are finite—fields such as  $\mathbf{Z}_p$ . As we'll see after a bit of investigation, there are other finite fields.

A **subfield** of a field  $F$  is a subset of  $F$  that is a field under  $F$ 's operators. If  $E$  is a subfield of  $F$ , then we say  $F$  is a **field extension** of  $E$ .

### M.7.1 Polynomials

One of the most useful ways of investigating fields is to consider polynomials  $c(x) = \sum_{0 \leq i \leq n} c_i x^i$  with **coefficients**  $c_i$  in some field. You probably remember this sort of thing from high school algebra. We define the **degree** of a non-zero polynomial to be the highest exponent of  $x$  having a non-zero coefficient. (This coefficient is called the **leading coefficient**.) By convention, the degree of the zero polynomial is  $-\infty$  (negative infinity). A polynomial of degree less than 1 is called a **constant**. A polynomial with leading coefficient 1 is called **monic**.

We define polynomial addition and multiplication in the usual way:

$$\sum_i c_i x^i + \sum_i d_i x^i = \sum_i (c_i + d_i) x^i$$

$$\sum_i c_i x^i \cdot \sum_i d_i x^i = \sum_i \sum_{j=0}^i c_{i-j} d_j x^i$$

Note that when multiplying polynomials, the degrees add. (See Homework Problem 14 for an example.) And that's why we had to define the degree of the zero polynomial to be  $-\infty$ .

We can also note that polynomials satisfy properties A, C, and D, *i.e.*, if  $a(x)$ ,  $b(x)$ , and  $c(x)$  are polynomials over some field, then property A:  $a(x) + (b(x) + c(x)) = (a(x) + b(x)) + c(x)$  and  $a(x)(b(x)c(x)) = (a(x)b(x))c(x)$ , property C:  $a(x) + b(x) = b(x) + a(x)$  and  $a(x)b(x) = b(x)a(x)$ , and property D:  $a(x) \cdot (b(x) + c(x)) = a(x)b(x) + a(x)c(x)$ . (See Homework Problem 15.)

Polynomial division is a bit more complicated, but suffice it to say that if we have two polynomials  $p(x)$  and  $d(x)$ , with  $d(x)$  non-zero, there are unique polynomials  $q(x)$  and  $r(x)$ , called the **quotient** and **remainder**, respectively, such that  $r(x)$  has degree smaller than  $d(x)$ , and  $p(x) = q(x)d(x) + r(x)$ . If  $r(x)$  is zero, we say that  $d(x)$  is a **factor** of  $p(x)$ , or that  $d(x)$  **divides**  $p(x)$ . There is

a simple algorithm, which we're sure you learned in high school algebra, for computing polynomial quotients and remainders (it's a lot like integer long division):

- Initialize  $q(x) \leftarrow 0$ ,  $r(x) \leftarrow p(x)$ .
- Repeat:  
 If  $\text{degree}(r(x)) < \text{degree}(d(x))$ , terminate.  
 $s(x) \leftarrow (\text{leading\_coefficient}(r(x))/\text{leading\_coefficient}(d(x)))x^{\text{degree}(r(x))-\text{degree}(d(x))}$   
 $q(x) \leftarrow q(x) + s(x)$   
 $r(x) \leftarrow r(x) - s(x)d(x)$

Sample Polynomial Division		
(polynomials over $\mathbf{Z}_5$ )		
divisor $\rightarrow 2x^3 + 3x + 1$	$  \begin{array}{r}  3x^2 + 3x + 1 \leftarrow \text{quotient} \\  \overline{) x^5 + x^4 + x^3 + x^2 + x + 1 \leftarrow \text{dividend}} \\  \underline{x^5 \phantom{+ 4x^3 + 3x^2}} \\  x^4 + 2x^3 + 3x^2 + x + 1 \\  \underline{x^4 \phantom{+ 4x^2 + 3x}} \\  2x^3 + 4x^2 + 3x + 1 \\  \underline{2x^3 \phantom{+ 3x + 1}} \\  4x^2 \phantom{+ 3x + 1} \leftarrow \text{remainder}  \end{array}  $	

(See Homework Problem 16.)

Just as we did with integers in §M.5 *Modular Arithmetic*, we can do modular arithmetic with polynomials. Given a non-zero polynomial  $m(x)$  as the **modulus**, we represent each polynomial  $p(x)$  by its remainder when divided by  $m(x)$ , writing  $p(x) \bmod m(x)$  for that remainder. And just as with integers, we can do addition and multiplication mod  $m(x)$  by doing regular addition and multiplication and then taking the remainder. (See Homework Problem 17.) And we can sometimes do division mod  $m(x)$ —by use of multiplicative inverses when they exist.

We can define the **greatest common divisor (gcd)** of two polynomials as the monic polynomial of highest degree that divides both of them. To compute the gcd, we can perform the Euclidean algorithm on polynomials in exactly the same way as we did for integers. We have to perform one final step—we divide by the leading coefficient of the last non-zero remainder to make the gcd monic. So, to compute the gcd of  $a(x)$  and  $b(x)$ , together with  $u(x)$  and  $v(x)$  such that  $\text{gcd}(a(x), b(x)) = u(x)a(x) + v(x)b(x)$ :

- Initial setup:  
 $r_{-2}(x) \leftarrow a(x) \quad u_{-2}(x) \leftarrow 1 \quad v_{-2}(x) \leftarrow 0$

$$r_{-1}(x) \leftarrow b(x) \quad u_{-1}(x) \leftarrow 0 \quad v_{-1}(x) \leftarrow 1 \\ n \leftarrow 0$$

- Step  $n$ :

If  $r_{n-1}(x) = 0$ , go to final step.

Otherwise, divide  $r_{n-2}(x)$  by  $r_{n-1}(x)$  to get quotient  $q_n(x)$  and remainder  $r_n(x)$ .

$$u_n(x) \leftarrow u_{n-2}(x) - q_n(x)u_{n-1}(x) \quad v_n(x) \leftarrow v_{n-2}(x) - q_n(x)v_{n-1}(x)$$

$$n \leftarrow n+1$$

Repeat.

- Final step:

$$t \leftarrow (\text{leading\_coefficient}(r_{n-2}(x)))^{-1}$$

$$\gcd(a(x), b(x)) \leftarrow t \cdot r_{n-2}(x) \quad u(x) \leftarrow t \cdot u_{n-2}(x) \quad v(x) \leftarrow t \cdot v_{n-2}(x)$$

For example, if  $a(x) = 2x^4 + 3x^3 + 4x^2 + 2x + 1$  and  $b(x) = 4x^3 + 1$  (polynomials over  $\mathbf{Z}_5$ ):

$n$	$q_n(x)$	$r_n(x)$	$u_n(x)$	$v_n(x)$
-2		$2x^4 + 3x^3 + 4x^2 + 2x + 1$	1	0
-1		$4x^3 + 1$	0	1
0	$3x + 2$	$4x^2 + 4x + 4$	1	$2x + 3$
1	$x + 4$	0	$4x + 1$	$3x^2 + 4x + 4$
2	$t = 4$	$\gcd = x^2 + x + 1$	$u(x) = 4$	$v(x) = 3x + 2$

We can **evaluate** a polynomial  $p(x)$  at a value  $v$  in the field by substituting the value  $v$  for each occurrence of  $x$  and doing the arithmetic (in the field, of course). Because of the way polynomial addition and multiplication are defined, any equation involving polynomial addition and multiplication will continue to be valid when all the polynomials are evaluated at any particular value  $v$ .

We define  $v$  to be a **root** of the polynomial  $p(x)$  if  $p(v) = 0$ . It's not hard to see that  $v$  will be a root iff the degree-1 polynomial  $(x-v)$  is a factor of  $p(x)$ :  $p(x) = q(x)(x-v) + r(x)$  with  $r(x)$  a constant, say,  $k$ , by polynomial division, so  $p(v) = q(v)(v-v) + k = k$ . Since a degree  $n$  polynomial can have at most  $n$  degree-1 factors, it can have at most  $n$  roots. (See Homework Problem 23.)

## M.7.2 Finite Fields

If  $F$  is a finite field with  $q$  elements, then every element of  $F$  is a root of  $x^q - x$ . Why? Well  $x^q - x = x \cdot (x^{q-1} - 1)$ , so clearly 0 is a root. Now, remember that  $F - \{0\}$  is a group under multiplication, and it has order  $q-1$ . Since the order of any group element divides the order of the group, each  $a \in F - \{0\}$

satisfies  $a^{q-1} = 1$ . So each non-zero element is a root of  $x^{q-1} - 1$ . Since there are  $q$  elements of  $F$ , and  $x^q - x$  has degree  $q$ , we can conclude that  $x^q - x = \prod_{a \in F} (x - a)$ .

In fact,  $F - \{0\}$  is cyclic. Why? Consider the order of each element. We know there is some element  $g$  whose order is the least common multiple  $\lambda$  of all those orders. We know that each of the  $q-1$  elements  $a$  in  $F - \{0\}$  satisfies  $a^\lambda = 1$  (because  $\lambda$  is a multiple of the order of  $a$ ), and so are roots of  $x^\lambda - 1$ . But  $x^\lambda - 1$  has degree  $\lambda$ , so there can be at most  $\lambda$  roots. Thus  $\lambda \geq q-1$ . And we know  $\lambda$  divides  $q-1$ , so  $\lambda = q-1$ , and  $F - \{0\}$  is cyclic with generator  $g$ .

### M.7.2.1 What Sizes Can Finite Fields Be?

For a finite field, if we start at 0 and continually add 1, we must eventually get back to 0. The number of times we can add 1 before getting back to 0 is called the **characteristic** of the field. Notationally, we'll write the sequence as  $0, 1, 2, \dots$ . Property D (distributivity) allows us to conclude that the characteristic is a prime: if the characteristic were  $ab$  with each of  $a$  and  $b$  smaller than the characteristic, then  $ab = (\sum_{a \text{ times}} 1)(\sum_{b \text{ times}} 1) = \sum_{ab \text{ times}} 1 = 0$ ; so  $F - \{0\}$  would contain  $a$  and  $b$  but not  $ab$ . Distributivity also allows us to conclude that if  $F$  has characteristic  $p$ , then for any  $c \in F$ ,  $\sum_{p \text{ times}} c = (\sum_{p \text{ times}} 1)c = 0 \cdot c = 0$ .

Now we'll enumerate the elements of a finite field  $F$  of characteristic  $p$ . We'll pick a sequence of elements  $a_i$  of  $F$  so that each new element can't be expressed as a linear combination  $\sum c_i a_i$  (with coefficients  $c_i \in \mathbf{Z}_p$ ) of the previously selected elements. Eventually, we'll run out of new elements. Then each element of  $F$  will be representable as a linear combination of elements of the sequence, with coefficients in  $\mathbf{Z}_p$ . If there are  $n$  elements in the sequence, then there are  $p^n$  linear combinations  $\sum_{i=1}^n c_i a_i$  where  $c_i \in \mathbf{Z}_p$ . No two of these can be equal because we could then solve for the last  $a_i$  for which the coefficients differ, as a linear combination of the earlier  $a_i$ s, contrary to our choice of  $a_i$ s.

$$\text{So } |F| = p^n.$$

### M.7.2.2 Representing a Field

It turns out that, for a given prime  $p$  and positive integer  $n$ , there is exactly one field of order  $q = p^n$ . It is the **splitting field** of  $x^q - x$  (considered as a polynomial over  $\mathbf{Z}_p$ ), the smallest field extension of  $\mathbf{Z}_p$  in which  $x^q - x$  factors completely into degree 1 polynomials. (See Homework Problem 26.) It is called the **Galois field** of order  $q$ , written  $\text{GF}(q)$ . But there are many ways to represent this field. We now have a sufficient handle on finite fields to be able to represent them in a way that allows us to compute with them.\* We'll choose a prime  $p$  and a positive integer  $n$ , thus determining the field  $F$  of order  $q = p^n$ . Pick a (multiplicative) generator  $g$  of  $F - \{0\}$ . Consider  $g^0, g^1, \dots, g^{n-1}$ . These must be linearly independent over  $\mathbf{Z}_p$ , i.e., there's no non-trivial linear combination of them that

---

\*For those of you who prefer code to wordy explanations, see <https://github.com/ms0/pymath. particularly ffield.py>.

equals 0, because otherwise we'd have  $g^k = \sum_{i < k} c_i g^i$  for some  $k < n$ , and so, since every non-zero element of  $F$  is  $g^m$  for some  $m$ , we could express every element of  $F$  as a linear combination of  $g^0, g^1, \dots, g^{k-1}$  with coefficients from  $\mathbf{Z}_p$ , but there are only  $p^k$  such linear combinations. Conversely, by the proof of  $|F| = p^n$ , there can be no more than  $n$  linearly independent elements of  $F$ , so  $g^n = \sum_{i < n} c_i g^i$  for some (unique) set of coefficients  $c_0, \dots, c_{n-1}$  determined by the generator we chose.

So we can represent any element  $a$  of  $F$  as a sequence of  $n$  elements of  $\mathbf{Z}_p$ , namely the coefficients  $a_i$  of  $g^0, g^1, \dots, g^{n-1}$  for which  $a = \sum_{i < n} a_i g^i$ . Addition is just componentwise addition in  $\mathbf{Z}_p$ . Multiplication is like polynomial multiplication, but we have to convert terms with exponents  $\geq n$  by use of  $g^n = \sum_{i < n} c_i g^i$ . The most straightforward technique (but by no means the most efficient) is to decompose the multiplication into a sequence of multiplications by  $g$ , multiplications by elements of  $\mathbf{Z}_p$ , and componentwise additions. Multiplying by  $g$  is easy:  $g \sum_{i < n} a_i g^i = \sum_{i < n} (a_{i-1} + c_i a_n) g^i$ , where  $a_{-1} = 0$ . Multiplication by  $b_j$  is trivial:  $b_j \sum_{i < n} a_i g^i = \sum_{i < n} a_i b_j g^i$ . So to multiply  $a = \sum_{i < n} a_i g^i$  by  $b = \sum_{j < n} b_j g^j$ , we multiply  $a$  by  $b_{n-1}$ , then for  $j = n-2$  to 0, multiply the result by  $g$  and add  $ab_j$ .

We also need to be able to compute negation (additive inverse) and (multiplicative) inverse. Negation is just componentwise negation in  $\mathbf{Z}_p$ . Inverse is more difficult, but as we commented before, we can make use of the Euclidean algorithm for polynomials over  $\mathbf{Z}_p$ . (Alternatively, we could exponentiate to the power  $q-2$ , since for  $a \in F - \{0\}$ ,  $a \cdot a^{q-2} = a^{q-1} = 1$ .) If  $a = \sum_{i < n} a_i g^i$ , let  $\alpha(x) = \sum_{i < n} a_i x^i$ . Let  $\gamma(x) = x^n - \sum_{i < n} c_i x^i$ . Note  $\gamma(g) = 0$ . If we perform the Euclidean algorithm on  $\alpha(x)$  and  $\gamma(x)$ , we get polynomials  $\mu(x)$  and  $\nu(x)$  such that  $\alpha(x)\mu(x) + \gamma(x)\nu(x) = \gcd(\alpha(x), \gamma(x))$ . Note that, since  $\gamma(g) = 0$ ,  $\alpha(g)\mu(g) = \gcd(\alpha(x), \gamma(x))(g)$ . If the gcd of  $\alpha(x)$  and  $\gamma(x)$  is 1, then  $\mu(g)$  is  $a^{-1}$ . But if the gcd is not 1, then  $a$  has no inverse. Since  $F - \{0\}$  is a group under multiplication, every nonzero  $a$  has an inverse, so no non-constant  $\alpha(x)$  can divide  $\gamma(x)$ , i.e.,  $\gamma(x)$  has no non-trivial factors. A polynomial with no non-trivial factors is called **irreducible**. So the field can be thought of as polynomials over  $\mathbf{Z}_p$  modulo the irreducible polynomial  $\gamma(x)$ .

We could equally well have started with the irreducible polynomial  $\gamma(x)$  instead of the element  $g$ .  $\gamma(x)$  is not just irreducible, but **primitive**, meaning that  $x$  has order  $p^n - 1 \bmod \gamma(x)$ . But any irreducible  $n$ th degree polynomial over  $\mathbf{Z}_p$  will produce a representation of  $\text{GF}(p^n)$  (see Homework Problem 25).

With this representation, an element of  $\text{GF}(2^n)$  could be represented as an  $n$ -bit binary number, with the least significant bit (little-endian bit 0) being the constant term of a  $\text{GF}(2)$  polynomial, bit 1 being the coefficient of  $x$ , and in general bit  $k$  being the coefficient of  $x^k$ . Addition is just bit-wise  $\oplus$ , while multiplication by  $x$  is just a left shift followed, if a bit gets shifted off the end, by an  $\oplus$  of the representation of  $\gamma(x) - x^n$  (see Figure M-2).

There is a slightly different way to represent a finite field of size  $p^n$  if  $n$  is composite, say,  $n = ab$ . We can choose as a modulus a monic degree  $a$  irreducible polynomial over  $\text{GF}(p^b)$ , say  $m(x)$ , using any representation of  $\text{GF}(p^b)$  for the coefficients. Then the elements of  $\text{GF}(p^n)$  are polynomials over  $\text{GF}(p^b) \bmod m(x)$ . This idea can be applied recursively, so, for example,  $\text{GF}(2^{2^k})$  elements

can be represented with  $2^k$  bits with each bit representing an element of  $\text{GF}(2)$ , then each pair of bits representing an element of  $\text{GF}(4)$ , then each pair of pairs of bits an element of  $\text{GF}(16)$ , and so on. In this representation, addition is still bitwise  $\oplus$ , while multiplication is done recursively. Rainbow [DING05] uses this representation.

+	0	1	2	3	4	5	6	7	×	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7	0	0	0	0	0	0	0	0	0
1	1	0	3	2	5	4	7	6	1	0	1	2	3	4	5	6	7
2	2	3	0	1	6	7	4	5	2	0	2	4	6	3	1	7	5
3	3	2	1	0	7	6	5	4	3	0	3	6	5	7	4	1	2
4	4	5	6	7	0	1	2	3	4	0	4	3	7	6	2	5	1
5	5	4	7	6	1	0	3	2	5	0	5	1	4	2	7	3	6
6	6	7	4	5	2	3	0	1	6	0	6	7	1	5	3	2	4
7	7	6	5	4	3	2	1	0	7	0	7	5	2	1	6	4	3

**Figure M-2.**  $\text{GF}(2^3)$  Arithmetic, Using Irreducible  $\mathbf{Z}_2$  Polynomial  $x^3+x+1$

## M.8 MATHEMATICS OF RIJNDAEL

Rijndael, the source of AES, is based on arithmetic in  $\text{GF}(2^8)$ . The nice thing about this is that each element of the field can be represented by an octet. The bits in this octet are the coefficients of a polynomial over  $\mathbf{Z}_2$  modulo the irreducible  $\mathbf{Z}_2$  polynomial  $m(x) = x^8+x^4+x^3+x+1$  (see Homework Problem 18), with the least significant bit being the constant coefficient and the most significant bit the coefficient of  $x^7$ . Addition in this representation is just bitwise  $\oplus$ , since addition modulo 2 is the same as  $\oplus$ . Also note that each element is its own negation. Multiplicative inverse is most easily done by table lookup. The multiplicative inverse table has only 256 1-octet entries (well, really 255, since 0 doesn't have an inverse). (See Homework Problem 19.) A multiplication table would have 65536 1-octet entries—one entry per pair of octets. Fortunately, Rijndael only requires multiplication by six different constants (elements of  $\text{GF}(2^8)$ ) not counting 0 and 1. Rijndael key expansion uses a sequence of constants  $C_i = x^{i-1} \bmod m(x)$ . (See Homework Problem 22.)

Rijndael also uses polynomials over  $\text{GF}(2^8)$ . These polynomials are taken modulo the  $\text{GF}(2^8)$  polynomial  $x^4+1$ . These polynomials are represented as 4-vectors of octets, with the constant coefficient being the first octet in the 4-vector. With this representation, multiplication by  $x$  is just a rotation, with the last octet becoming the first. Rijndael's *MixColumns* operation is multiplication by the fixed polynomial  $c(x) = 03x^3+01x^2+01x+02$ , which we'll call the **MixColumns polynomial**. *InvMixColumns* is multiplication by  $d(x) = 0Bx^3+0Dx^2+09x+0E$ , which we may as well call the **InvMixColumns polynomial**. (See Homework Problem 21.)

Finally, for one of the operations composing its S-box, Rijndael treats octets as polynomials over  $\mathbf{Z}_2$  modulo the non-irreducible  $\mathbf{Z}_2$  polynomial  $x^8+1$ , again with the least significant bit being



the constant coefficient and the most significant bit the coefficient of  $x^7$ . This S-box component consists of multiplying by  $x^4+x^3+x^2+x+1$  and then adding  $x^6+x^5+x+1$ . Its inverse consists of multiplying by  $x^6+x^3+x$  and then adding  $x^2+1$ . (See Homework Problem 20.)

In fact, Rijndael's S-box is composed of a sequence of three invertible operations in which octets are interpreted as  $\mathbf{Z}_2$  polynomials:

1. a permutation in which each octet maps to its multiplicative inverse mod  $m(x)$  (except for 0 which maps to itself)
2. multiplication by  $x^4+x^3+x^2+x+1 \bmod x^8+1$
3. addition of  $x^6+x^5+x+1$

Rijndael's *MixColumns* is just multiplication by the *MixColumns* polynomial  $c(x) \bmod x^4+1$ , and its inverse is just multiplication by the *InvMixColumns* polynomial  $d(x) \bmod x^4+1$ .

The rotation of the columns performed in Rijndael's key expansion can be thought of as multiplication by  $x^3 \bmod x^4+1$ .

### M.8.1 A Rijndael Round

The actual description of a Rijndael round, as found in the spec, is as follows. Recall that each round operates on a state consisting of  $N_b$  4-octet columns. (Note that  $N_b=4$  for AES.)

1. The S-box is applied to each octet in the state.
2. Row 0 (the top row) of the state is left alone.  
Row 1 of the state is rotated left 1 column.  
Row 2 of the state is rotated left  $2 + \lfloor N_b/8 \rfloor$  columns (2 if  $N_b < 8$ , 3 otherwise).  
Row 3 of the state is rotated left  $3 + \lfloor N_b/7 \rfloor$  columns (3 if  $N_b < 7$ , 4 otherwise).
3. *MixColumns* is applied to each column.
4. The round key is  $\oplus$ 'd into the state.

We can think of the initial  $\oplus$  as an initial round (round 0) which only has step 4. The final round (round  $N_r$ ) is missing step 3.

You might think that the inverse of encryption would just run encryption backwards, with each operation replaced by its inverse. And this, of course, works fine. But by regrouping and reordering the operations, decryption can look much more like encryption. The key observations for this are that steps 1 and 2 can be exchanged without changing the result, and that *InvMixColumns* is polynomial multiplication, and so it distributes over addition ( $\oplus$ ): If  $s(x)$  is a column of the state, and  $k(x)$  is the corresponding column of the round key, and  $d(x)$  is the *InvMixColumns* polynomial, then  $(s(x) \oplus k(x)) \cdot d(x) = s(x)d(x) \oplus k(x)d(x)$ , and, of course, this stays true mod  $x^4+1$ .

So if we group steps 3 and 4 from one round with steps 1 and 2 from the next, then interchange steps 1 and 2, and interchange steps 3 and 4 by applying *InvMixColumns* to each column of

the round key, then, when we run this backwards with the operations inverted, we get the following description of an inverse round.

1. The Inverse S-box is applied to each octet in the state.
2. Row 0 (the top row) of the state is left alone.  
 Row 1 of the state is rotated right 1 octet.  
 Row 2 of the state is rotated right  $2 + \lfloor N_b/8 \rfloor$  columns (2 if  $N_b < 8$ , 3 otherwise).  
 Row 3 of the state is rotated right  $3 + \lfloor N_b/7 \rfloor$  columns (3 if  $N_b < 7$ , 4 otherwise).
3. *InvMixColumns* is applied to each column.
4. The (modified) round key is  $\oplus$ 'd into the state.

This is the way decryption rounds 1 through  $N_r - 1$  work. Decryption round 0 is just the inverse of step 4 from encryption round  $N_r$  (remember that there is no step 3 in that round—now you know why), namely  $\oplus$  of the (unmodified) round key. Decryption round  $N_r$  is the inverse of step 4 (the only step) of encryption round 0 followed by steps 1 and 2 from encryption round 1. So it is missing inverse step 3 (there's the other side of the reason for the final round being the way it is), and its round key is unmodified (*i.e.*, it's just encryption round key 0).

## M.9 ELLIPTIC CURVE CRYPTOGRAPHY

An elliptic curve is a set of points on the coordinate plane satisfying an equation of the form  $y^2 = x^3 + ax^2 + bx + c$ . If you have two points on such a curve and draw a line through both of them, the line will intersect the curve at a unique third point, because in a field, if a polynomial of degree 3 has two roots (say,  $r$  and  $s$ ), you can divide the polynomial by the product  $(x-r)(x-s)$ , and you end up with a degree-1 polynomial from which you can read off the third root. (There are a couple of special cases. If the two points are the same, the line we draw is the tangent to the curve. If the two points are opposite each other about the  $x$  axis, *i.e.* their abscissas ( $x$  coordinates) are the same and their ordinates ( $y$  coordinates) are additive inverses, the third point is at infinity, and we'll call it **I**.)

So you can define a weird kind of multiplication\* of points on an elliptic curve. To get the product of two points, draw a line through the two points, find the third intersection point of the curve and the line, and reflect that point across the  $x$  axis (by negating its ordinate). What's so amazing about this multiplication is that it is associative! We just have to define the inverse of a point as the opposite point about the  $x$  axis, and then **I** is the identity, and we have a group.

---

\*Typical descriptions of elliptic curve groups show the group operation as addition and the identity as **O**. As a consequence, exponentiation is instead written as left-multiplication by an integer. We've chosen to use the multiplicative paradigm to emphasize the similarity with cryptography based on  $\mathbf{Z}_p^\times$ , such as Diffie-Hellman.

If our coordinate plane is not the real plane, but instead the coordinate plane of a finite field, then this elliptic curve group is finite. One can then pick an element from this group and use the resulting cyclic subgroup for cryptography. For example, Diffie-Hellman works for this group the same way it does for  $\mathbf{Z}_p^\times$ . The discrete logarithm problem for elliptic curve groups appears to be harder than for  $\mathbf{Z}_p^\times$ , at least for a conventional computer.

In common usage, coefficient  $a$  is chosen to be zero. There is no loss of generality in doing this, because  $a$  is minus the sum of the roots of the cubic, so a horizontal translation of the curve by a third of that sum puts it in the restricted form. (Of course, if the field has characteristic 3, you can't do that, but no one uses fields of characteristic 3 for elliptic curve cryptography.)

There is a slight complication for some of the (computationally) best finite fields, namely,  $\text{GF}(2^n)$ . Because these fields have characteristic 2,  $x = -x$  and so there is only one square root of each square, which pretty much wrecks the scheme described above. But all is not lost. If we skew the coordinate system by a linear transformation so that each point  $\langle x, y \rangle$  becomes  $\langle x, y - \frac{1}{2}x \rangle$ , then iff the original point satisfied  $y^2 = x^3 + ax^2 + bx + c$ , the new point satisfies  $(y + \frac{1}{2}x)^2 = x^3 + ax^2 + bx + c$  or  $y^2 + xy = x^3 + (a - \frac{1}{4})x^2 + bx + c$ . So this suggests that we can look at skewed elliptic curves of the form  $y^2 + xy = x^3 + ax^2 + bx + c$ , and points on these curves will have the same group properties. But over a field of characteristic two, we see that if  $\langle x, y \rangle$  is on such a curve, so is  $\langle x, y+x \rangle$ , because  $(y+x)^2 + x(y+x) = y^2 + x^2 + xy + x^2 = y^2 + xy$  (remember that for characteristic 2,  $1+1=0$ ). So this allows us to use  $\text{GF}(2^n)$ , which is one of the two common finite field categories used for elliptic curve cryptography. For  $\text{GF}(2^n)$ ,  $b$  (the coefficient of  $x$ ) is chosen to be 0, whereas for the other common choice ( $\mathbf{Z}_p$ , for large primes  $p$ ),  $a$  (the coefficient of  $x^2$ ) is chosen to be 0.

*Some noncommutative groups follow!*

## M.10 RINGS

Remember, a **ring** is a structure  $\langle R, +, \times \rangle$  with two binary operators ( $+$  and  $\times$ ) that each satisfy property A (associativity), and together satisfy property D (distributivity), while  $\langle R, + \rangle$  is a commutative group whose identity element is usually written as 0. Examples of rings are

- any field
- $\mathbf{Z}$ , the integers
- $\mathbf{Z}_n$ , the integers mod  $n$
- $\mathbf{Z}[i]$ , the Gaussian integers (complex numbers with integer real and imaginary parts)
- $\mathbf{H}$ , the quaternions (discovered in 1843 by Irish mathematician William Rowan Hamilton)
- $R[x]$ , the polynomials in one variable with coefficients in ring  $R$
- $R[x]/p(x)$ , the polynomials over  $R$ , modulo the polynomial  $p(x)$
- $\text{Mat}_n(R)$ , the  $n \times n$  matrices with elements in ring  $R$ , described in §M.12

## M.11 LINEAR TRANSFORMATIONS

A linear transformation is a way to take an input, comprising a sequence (**vector**) of elements from a ring, and produce a new vector, the output, where each output element is a weighted sum of the input elements. So if there are  $m$  input elements  $a_i$  for\*  $1 \leq i \leq m$ , and  $n$  output elements  $b_j$  for  $1 \leq j \leq n$ , a general linear transformation can be represented as a matrix  $\mathbf{W}$  of  $m \times n$  weights  $w_{ij}$  for  $1 \leq i \leq m$  and  $1 \leq j \leq n$ . We write  $\mathbf{W}$  as a rectangular array of weights with  $m$  rows and  $n$  columns:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

If we write the input as a  $1 \times m$  matrix  $\mathbf{A} = [a_1 \ a_2 \ \dots \ a_m]$  and the output as a  $1 \times n$  matrix  $\mathbf{B} = [b_1 \ b_2 \ \dots \ b_n]$ , then we can write  $\mathbf{AW} = \mathbf{B}$ , with the rule  $\sum_{1 \leq i \leq m} a_i w_{ij} = b_j$ . It's easy to see that if we apply two linear transformations  $\mathbf{W}$  and  $\mathbf{X}$  in succession, we get another linear transformation—after all, a weighted sum of weighted sums is still a weighted sum. And it's not too hard to see that to combine  $\mathbf{W}$  and  $\mathbf{X}$ , written as matrix multiplication  $\mathbf{Y} = \mathbf{WX}$ , the rule is  $y_{ij} = \sum_{1 \leq k \leq n} w_{ik} x_{kj}$ . (See Homework Problem 28.) Note that for it to be possible to multiply two matrices, the number of columns in the first has to be the same as the number of rows in the second, and the product will have the same number of rows as the first and the same number of columns as the second.

$$\begin{bmatrix} \vdots & \vdots & \vdots \\ a_{i1} & \cdots & a_{ik} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \cdots & b_{1j} & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & b_{kj} & \cdots \end{bmatrix} = \begin{bmatrix} \ddots & & \vdots & & \ddots \\ \cdots & c_{ij} = a_{i1}b_{1j} + \cdots + a_{ik}b_{kj} & \cdots \\ \ddots & & \vdots & & \ddots \end{bmatrix}$$

**Figure M-3.** Matrix Multiplication

We can also add or subtract pairs of matrices, simply by adding or subtracting corresponding elements. In this case, only matrices of the same **dimensions**, *i.e.*, the same numbers of rows and columns, can be added or subtracted.

Some people prefer to put the weights before the vector:  $\mathbf{W}'\mathbf{A}' = \mathbf{B}'$ . In order to make this work without changing how we multiply, we **transpose** the matrices—that is, we flip them over diagonally by interchanging their dimensions. So a  $1 \times m$  **row vector** is transposed to an  $m \times 1$  **column vector** and an  $m \times n$  matrix is transposed to an  $n \times m$  matrix: If  $\mathbf{W}_{ij} = w_{ij}$ ,  $\mathbf{W}$ 's transpose  $\mathbf{W}^T$  has

---

\*As is the usual mathematical convention, we use indices starting at 1. Computational implementations often use indices starting at 0.

$(\mathbf{W}^T)_{ij} = w_{ji}$ . It is trivial to verify that if  $\mathbf{Y} = \mathbf{W}\mathbf{X}$ , then  $\mathbf{Y}^T = \mathbf{X}^T \mathbf{W}^T$ , provided that multiplication of elements is commutative. (See Homework Problem 29.)

$$\mathbf{C} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \mathbf{C}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

**Figure M-4.** Matrix Transpose

## M.12 MATRIX ARITHMETIC

It is not hard to verify that matrix multiplication is associative and that it distributes over matrix addition. (See Homework Problem 30.) Matrix multiplication is not, however, commutative, even for same-sized **square matrices**, where the number of rows is the same as the number of columns so that multiplying them in either order is possible. (See Homework Problem 31.) But for each size  $n$ , there is an  $n \times n$  **identity matrix**  $\mathbf{I}_n$  which, when multiplied with any compatibly dimensioned matrix results in that same matrix. It has 1s along its main diagonal and 0s everywhere else. Its elements can be written using the **Kronecker delta**  $\delta_{ij} = (1 \text{ if } i = j; 0 \text{ otherwise})$ . (See Homework Problem 32.) When it's clear from context, we'll drop the subscript.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

For each  $m \times n$ , there's also a **zero matrix**  $\mathbf{0}_{mn}$  all of whose elements are zero. Again, we'll normally drop the subscripts.

$$\mathbf{0} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

You may have noticed that we haven't said anything yet about matrix division. We can sometimes perform division by a square matrix  $\mathbf{M}$  by finding its **inverse**  $\mathbf{M}^{-1}$ , a matrix such that  $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$ , and just multiplying by  $\mathbf{M}^{-1}$ . Since any matrix represents a linear transformation, and

since not every linear transformation can be inverted, *e.g.*, suppose an entire column of weights is zero, it is clear that only some matrices can have inverses.

The astute reader may have noticed that we've described  $M^{-1}$  as a *right* inverse. It is not immediately obvious that  $M^{-1}M = I$ , *i.e.*, that  $M$  is a right inverse of  $M^{-1}$ . But associativity guarantees that if  $M^{-1}$  has a right inverse, it is  $M$ , and that  $M$  has only one right inverse. (See Homework Problem 34.) §M.12.2 *Matrix Inverses* will resolve the remaining question. Still, when using inverses for division, it matters whether you multiply on the left or the right. (See Homework Problem 36.)

What about inverses of non-square matrices? While one-sided inverses can exist, they are not unique, and they can never be two-sided because the underlying linear transformation changes the size of the vector. So for the rest of the document, we'll be talking about square matrices.

### M.12.1 Permutations

There is a particular kind of linear transformation that just rearranges the elements of the input vector to produce the output vector. The weights are all 0 except for the 1s for each *⟨input element position, corresponding output position⟩* pair. The equivalent **permutation matrix** has exactly one 1 in each row and in each column, with all other elements being 0. It's easy to see that the transpose of a permutation matrix is also its inverse. (See Homework Problem 39.)

### M.12.2 Matrix Inverses

In this section, we give an algorithm (**Gaussian elimination**) for computing the inverse of a square matrix over a division ring  $R$ , or determining that it has no inverse. The basic idea is to attempt to slowly transform the matrix  $M$  into the identity matrix by multiplying on the left by a sequence of simple invertible matrices, so  $I = X_k X_{k-1} \cdots X_2 X_1 M$ . Then clearly  $X_k X_{k-1} \cdots X_2 X_1$  is a left inverse of  $M$ . Furthermore  $X_k X_{k-1} \cdots X_2 X_1$  has a two-sided inverse, namely  $X_1^{-1} X_2^{-1} \cdots X_{k-1}^{-1} X_k^{-1}$ , which therefore must be  $M$  (see Homework Problem 34). Each  $X$  will perform one of three transformations when applied on the left— $S_{rs}$ : swap rows  $r$  and  $s$ ,  $T_{m,r}$ : multiply row  $r$  by a nonzero element  $m$  of  $R$ , or  $A_{m,rs}$ : add  $m$  times row  $r$  to row  $s \neq r$ . Note that each of these has an inverse of the same type:  $S_{rs}^{-1} = S_{rs}$ ,  $T_{m,r}^{-1} = T_{1/m,r}$ , and  $A_{m,rs}^{-1} = A_{-m,rs}$ . Homework Problem 37 asks for the matrices corresponding to these three transformations.

#### M.12.2.1 Gaussian Elimination

This algorithm, named for the mathematician Carl Friedrich Gauss, computes the inverse of a square matrix. We've written it in Python style, but to be consistent with the 1-based indices that

we've used throughout this document, we use `Range` instead of `range`, where `Range(n)` starts at 1 and ends at `n`, and `Range(m,n)` starts at `m` and ends at `n`.

```

append an n by n identity matrix to the right of M, making M n by 2n
for c in Range(n):          # for each column c, 1 through n
    for r in Range(c,n):    # look in row c and rows below
        if M[r,c]: break    # until find a nonzero element in column c
    else:                   # no suitable row found
        raise ZeroDivisionError('not invertible')
    apply S(r,c)            # swap the suitable row with row c
    apply T(1/M[c,c],c)     # make the diagonal element 1
    for r in Range(c+1,n):  # for each row below row c
        apply A(-M[r,c],c,r) # zero out column c
for c in Range(2,n):        # for each column c, 2 through n
    for r in Range(c-1):    # for each row above row c
        apply A(-M[r,c],c,r) # zero out column c

```

At the completion of this algorithm, the left half of **M** (columns 1 through `n`) is an identity matrix and the right half of **M** (columns `n+1` through `2n`) is the inverse of the input matrix, provided that the input matrix is invertible (see Homework Problem 38).

## M.13 DETERMINANTS

For the remainder of this document, we will assume that the matrix elements are members of a commutative ring with identity. Without mentioning it, we've already been assuming the elements are members of a ring with identity, but commutativity is about to become important as well. Note that every field is a commutative ring with identity, as is  $\mathbf{Z}_n$  even when  $n$  is not prime. As with fields, we'll refer to additive inverses as **negations** and multiplicative inverses simply as **inverses**.

The determinant of a square matrix is a magic quantity with magical properties. For an  $n \times n$  matrix **A**, it's defined as

$$\det(\mathbf{A}) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^n A_{i,\pi(i)}$$

In English, it's the sum of all terms where each term takes one element from each row and each column and multiplies them together and negates the product if the sequence of columns chosen from successive rows is an odd permutation of  $1, \dots, n$  (see Homework Problem 8). As you can see, there are  $n!$  terms, where each term is the (sometimes negated) product of  $n$  elements, but luckily there are more efficient ways of computing the determinant. (See Homework Problem 40.)

### M.13.1 Properties of Determinants

It should be reasonably obvious that the determinant of the transpose of a matrix is equal to the determinant of the matrix itself, because we get the same terms in both cases, just in a different order. Even more obvious is that the determinant of a **diagonal matrix**, which is a matrix whose only nonzero elements lie along its upper-left to lower-right diagonal, is just the product of the elements on that diagonal, since any permutation other than the identity selects a zero element. In fact, the determinant of a **triangular matrix**, which is a matrix all of whose elements below (or above) its diagonal are zero, is the product of the elements on its diagonal, for the same reason.

What is nearly as obvious is that, for any matrix, if you multiply each element of a row (or a column) by a constant, you multiply the determinant by that constant. Less obvious is that if you add a multiple of one row (or column) to another, the determinant doesn't change. This is actually a consequence of the observation that a matrix with two rows or columns the same has determinant 0, as pairs of permutations produce identical terms except for their sign, resulting in complete cancellation. Finally, swapping two rows (or columns) of a matrix negates its determinant—the swap reverses the parity of each permutation.

The above facts suggest an efficient algorithm for computing the determinant by cleverly adding multiples of rows to other rows in order to produce a triangular matrix. (See Homework Problem 40.)

The most magical property of determinants is that  $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$ . We prove this in §M.13.2.

#### M.13.1.1 Adjugate of a Matrix

We define the adjugate of a matrix by  $(\text{adj}(\mathbf{M}))_{ji} = (-1)^{i+j}\det(\mathbf{M} \text{ with row } i \text{ and column } j \text{ removed})$ . It is not hard to show that  $\text{adj}(\mathbf{M})\mathbf{M} = \mathbf{M}\text{adj}(\mathbf{M}) = \det(\mathbf{M})\mathbf{I}$ , where multiplying a matrix with a single value means multiplying each element by that value. (See Homework Problem 41.)

If  $\mathbf{M}$  has an inverse  $\mathbf{M}^{-1}$ ,  $\det(\mathbf{M})\det(\mathbf{M}^{-1}) = \det(\mathbf{MM}^{-1}) = \det(\mathbf{I}) = 1$ , so  $\det(\mathbf{M})$  has an inverse. Conversely, if  $\det(\mathbf{M})$  has an inverse, then  $\mathbf{M}(\text{adj}(\mathbf{M})/\det(\mathbf{M})) = \mathbf{I} = (\text{adj}(\mathbf{M})/\det(\mathbf{M}))\mathbf{M}$ , where division is merely multiplication by the inverse. In other words, a matrix has an inverse iff its determinant has an inverse, in which case  $\mathbf{M}^{-1} = \text{adj}(\mathbf{M})/\det(\mathbf{M})$ . For matrices over fields, where inverses exist for all nonzero field elements, a matrix has an inverse iff its determinant is nonzero.



### M.13.2 Proof: Determinant of Product is Product of Determinants

A preliminary observation:

$$\prod_{l=1}^n \sum_{k=1}^m C_{l,k} = \sum_{k \in \{1, \dots, m\}^n} \prod_{l=1}^n C_{l,k_l}$$

In English: the product of equal-length sums is the sum of all the terms formed by multiplying together one term from each sum, *e.g.*,  $(a+b+c)(d+e+f) = ad+bd+cd+ae+be+ce+af+bf+cf$ . This is a straightforward consequence of distributivity. Notation:  $\{1, \dots, m\}^n$  is the set of  $n$ -tuples of elements of  $\{1, \dots, m\}$ .

And now the proof...

$$\begin{aligned} \det(\mathbf{AB}) &= \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{l=1}^n (\mathbf{AB})_{l, \pi(l)} \\ &= \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{l=1}^n \sum_{k=1}^n A_{l,k} B_{k, \pi(l)} \\ &= \sum_{k \in \{1, \dots, n\}^n} \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{l=1}^n A_{l, k_l} B_{k_l, \pi(l)} \end{aligned}$$

Now, here's where some massive cancellation happens. Each  $k$  is a sequence of  $n$  indices. If any two indices are the same, say  $k_p = k_q$ , then, in the sum over all permutations, the term for a particular permutation  $\pi$  contains the same factors from  $\mathbf{A}$  and  $\mathbf{B}$  as the term for that same permutation but with  $p$  and  $q$  first swapped. The only difference is that the sign of that permutation has been reversed by the swap, so the terms cancel! So only the  $k$ s that are permutations contribute to the sum, and we can replace the outer sum with a sum over permutations. From there, it's just a somewhat simple matter of rearranging terms, liberally applying associativity and distributivity:

$$\begin{aligned} \det(\mathbf{AB}) &= \sum_{k \in \{1, \dots, n\}^n} \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{l=1}^n A_{l, k_l} B_{k_l, \pi(l)} \\ &= \sum_{\kappa \in S_n} \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{l=1}^n A_{l, \kappa(l)} B_{\kappa(l), \pi(l)} \\ &= \sum_{\kappa \in S_n} \sum_{\rho \in S_n} \operatorname{sgn}(\rho \kappa^{-1}) \prod_{l=1}^n A_{l, \kappa(l)} \prod_{m=1}^n B_{m, \rho(m)} \\ &= \left( \sum_{\kappa \in S_n} \operatorname{sgn}(\kappa) \prod_{l=1}^n A_{l, \kappa(l)} \right) \left( \sum_{\rho \in S_n} \operatorname{sgn}(\rho) \prod_{m=1}^n B_{m, \rho(m)} \right) \\ &= \det(\mathbf{A}) \det(\mathbf{B}) \end{aligned}$$

## M.14 HOMEWORK

### *arithmetic*

1. Use properties A, I and N for addition and property D to show that  $0 \times a = 0 = a \times 0$  and  $(-a) \times b = -(a \times b) = a \times (-b)$ .
2. Derive rules for addition and multiplication of fractions using properties A, C, D, I, and N. In the process, show that  $a/b = (ac)/(bc)$  for any nonzero integer  $c$ , and conclude that  $a/b = (a/g)/(b/g)$  where  $g = \gcd(a, b)$ ; show that  $a/g$  and  $b/g$  are relatively prime.
3. If  $a$  and  $b$  are relatively prime and  $a \mid bc$ , show that  $a \mid c$ . Hint:  $1 = \gcd(a, b) = ua + vb$ .
4. Show that the Euler totient function  $\phi$  is multiplicative, *i.e.*, if  $a$  and  $b$  are relatively prime,  $\phi(ab) = \phi(a)\phi(b)$ . Hint: Use the Chinese remainder theorem.  
What is  $\phi(n)$  for  $n = 2^{\alpha_0} p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \cdots p_k^{\alpha_k}$  where  $p_i$  are distinct odd primes?
5. Suppose  $m = st$ , where  $s$  and  $t$  are  $b$ -bit coprimes. Let  $x$  be a  $2b$ -bit base and  $e$  a  $2b$ -bit exponent. Using multiplication and division algorithms that for  $k$ -bit numbers take time  $k^2$ , about how long would it take to calculate  $x^e \bmod m$  using the exponentiation algorithm from §M.5.3, doing modular reduction after each multiply (so that each  $k$ -bit modular multiply takes time  $2k^2$ ). What if, instead, you calculate  $x^e \bmod s$  and  $x^e \bmod t$  separately, and then use the Chinese remainder theorem (§M.5.2.2) to get  $x^e \bmod m$ ? Don't include the time for computing  $s^{-1} \bmod t$  and  $t^{-1} \bmod s$ , as those can be precomputed. What if you reduce the exponent  $\bmod \lambda(s)$  for the mod  $s$  exponentiation and  $\bmod \lambda(t)$  for the mod  $t$  exponentiation?

### *groups*

6. (A) Show that for any prime  $p$  and any  $x \in \mathbf{Z}_p$ ,  $x^{k\phi(p)+1} = x$ . Hint: It's clearly true for  $x = 0$ , and  $\langle \mathbf{Z}_p^\times, \times \rangle$  is a group whose order is  $\phi(p)$ . Note that  $\phi(p) = \lambda(p)$ .  
(B) Use the Chinese remainder theorem to show that if  $m$  is square-free and  $x \in \mathbf{Z}_m$ ,  $x^{k\lambda(m)+1} = x$ .
7. For an Abelian group  $\langle G, \cdot \rangle$ , show that  $\prod_{x \in G} x = \prod_{x \in G, x \cdot x = e} x$ . Conclude that  $(p-1)! + 1 = 0 \bmod p$  for prime  $p$ . Hint:  $\mathbf{Z}_p$  is a field.
8. A) A finite permutation,  $\pi$ , is a rearrangement of a finite sequence of objects, which we might as well number  $1, 2, \dots, n$ . If we have a permutation, we can look at where 1 goes ( $\pi(1)$ ), and then look at where the number at that position goes ( $\pi(\pi(1))$ ), and so forth until we get back to 1. Why do we have to get back to 1?  
B) We show this cycle as  $(1 \ \pi(1) \ \pi(\pi(1)) \ \dots)$ . (We don't bother listing 1 twice; the understanding is that the last listed element goes to the first.) We then look for the next object that's not in any of the cycles we've already listed, say  $k$ , and find its cycle  $(k \ \pi(k) \ \pi(\pi(k)) \ \dots)$ , and so forth until we have accounted for all the objects  $1, 2, \dots, n$ . As an optimization, we don't bother listing length-1 cycles; the understanding is that any object that is not listed is not

moved. But just so we have a notation for the identity permutation that doesn't move anything, we write it as  $()$ .

Some of the permutation of four objects are thus  $()$ ,  $(1\ 2)$ ,  $(1\ 2)(3\ 4)$ . List all the permutations of three objects.

C) Composition. As long as two permutations in this cycle notation don't share any objects, it doesn't matter what order we compose them in. But otherwise it does. And unfortunately, there are two ways to do it: left to right and right to left. Religious wars amongst mathematicians have been fought over this, much like big-endian and little-endian in the computer field. The problem is that if  $\pi$  and  $\rho$  are two permutations, we might want the permutation specified by  $\pi(\rho(k))$  to be the composition  $(\pi \cdot \rho)(k)$ . But this means we have to apply the cycles right to left, even though within a cycle the notation is clearly left to right. We could just redefine the cycle notation so that each successive object in the list tells from where the object immediately to its left came, and then it would make sense to read the cycles from right to left. Another approach, fairly common amongst algebraists, is to write arguments to the right of the function, as in  $(k)\pi$ . Then  $((k)\pi)\rho$  would be  $(k)(\pi \cdot \rho)$  with the cycles just concatenated. This is also consistent with multiplication of row vectors by square matrices. Anyway, for purposes of this book, we'll specify that  $(\pi \cdot \rho)$  means first apply  $\rho$  and then  $\pi$ . Write out the composition table for the permutations of three objects, and verify that this is a non-commutative group.

D) Show that composition of permutations is an associative operator. Conclude that the set of permutations of  $n$  objects forms a group under composition. What permutation is the identity? How would you convert a permutation written in the cycle notation to its inverse?

E) The group of permutations of  $n$  objects is called the **symmetric group** of  $n$  objects, commonly written  $S_n$ . What is the order of  $S_n$ ? Show that every element of  $S_n$  can be written as the composition of two-object cycles, otherwise known as **swaps**. Furthermore, even though there are many ways to do this, show that for any given permutation, the number of swaps used will always have the same parity, *i.e.*, it will always be even or always be odd. Hint: What happens to the cycles of a permutation when you swap two elements that are (a) in the same cycle? (b) in different cycles? What happens to the number of even-length cycles? How many even-length cycles are there in the identity permutation? The **parity** of a permutation is the parity of the number of swaps (which you have just shown is also the parity of the number of even-length cycles). The **sign** of a permutation  $\pi$ , written  $\text{sgn}(\pi)$ , is  $+1$  if  $\pi$  is even and  $-1$  if  $\pi$  is odd. Show that the sign of the composition of two permutations is the product of their signs. Since the identity permutation is even, conclude that the set of even permutations of  $n$  objects forms a subgroup of  $S_n$ . This subgroup is called the **alternating group** of  $n$  objects, commonly written  $A_n$ . What is the order of  $A_n$ ?

F) Show that every group is a subgroup of a permutation group. Hint: Multiplying by an element of the group permutes the elements (because then multiplying by that element's inverse restores the original order).

9. Show that in the permutation group of three objects, there are elements of orders 2 and 3, but no element of order  $\text{lcm}(2,3) = 6$ .
10. Show that the group of permutations of  $n > 2$  objects does not have a generator. But show that any permutation of  $n$  objects is the composition of a sequence of instances of the permutations  $(1\ 2)$  and  $(1\ 2\ \dots\ n)$ .
11. This is one of my<sub>3</sub> favorite problems. It shows the power of associativity. See §M.6 *Groups*. Consider the following properties:

- (LI) Existence of **left identity**. There exists an element  $e$  such that, for each  $a$ ,  $ea = a$ .
- (LN) Existence of **left inverse**. For each  $a$ , there is an  $a^{-1}$  such that  $a^{-1}a = e$ .
- (RI) Existence of **right identity**. There exists an element  $e$  such that, for each  $a$ ,  $ae = a$ .
- (RN) Existence of **right inverse**. For each  $a$ , there is an  $a^{-1}$  such that  $aa^{-1} = e$ .

Show that if  $\langle G, \cdot \rangle$  has properties A, LI, and LN (or properties A, RI, and RN), it is a group.

Nonhint: This would be trivial if you could assume property C. So don't!

Hint: Consider  $(a^{-1})^{-1}a^{-1}aa^{-1}$ .

Find a non-group with properties A, LI, and RN.

### *rings, polynomials, and fields*

12. Why isn't  $\mathbf{Z}_n$  a field when  $n$  isn't prime? Which property fails?
13. (A) Show that, for the reduced totient function  $\lambda$ , if  $a$  and  $b$  are relatively prime,  $\lambda(ab) = \text{lcm}(\lambda(a), \lambda(b))$ . Hint: Use the Chinese remainder theorem and the fact that if an Abelian group has an element of order  $\alpha$  and an element of order  $\beta$ , it has an element of order  $\text{lcm}(\alpha, \beta)$ .  
 (B) Show that for  $p$  an odd prime,  $\lambda(p^n) = (p-1)p^{n-1}$ . Hint: First show that it's true for  $n=1$ , noting that  $\mathbf{Z}_p$  is a field, so its multiplicative group is cyclic. Next, show that it's true for  $n=2$ , noting that if  $g$  is a generator of  $\mathbf{Z}_p^\times$ , either  $g$  or  $g+p$  is a generator of  $\mathbf{Z}_{p^2}^\times$ . Finally, show that if  $g$  is a generator of  $\mathbf{Z}_{p^2}^\times$ , it is a generator of  $\mathbf{Z}_{p^n}^\times$ .  
 (C) Show that  $\lambda(2) = 1$ ,  $\lambda(4) = 2$ , and  $\lambda(2^n) = 2^{n-2}$  for  $n > 2$ . Hint: 3 has maximal order in  $\mathbf{Z}_{2^n}^\times$  for  $n > 1$ , and  $(2k+1)^2 = 4k^2 + 4k + 1 = 8(k(k+1)/2) + 1$ .  
 (D) What is  $\lambda(n)$  for  $n = 2^{\alpha_0} p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \dots p_k^{\alpha_k}$  where  $p_i$  are distinct odd primes?  
 (E) If  $n = 2^{\alpha_0} p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \dots p_k^{\alpha_k}$  where  $p_i$  are distinct odd primes, show that, mod  $n$ , 1 has  $2^k$  square roots if  $\alpha_0 \leq 1$ ,  $2^{k+1}$  square roots if  $\alpha_0 = 2$ , and  $2^{k+2}$  square roots if  $\alpha_0 \geq 3$ . Hints: Use the Chinese remainder theorem to show that  $x$  is a square root of 1 mod  $n$  iff it is a square root of 1 mod each of  $x$ 's prime power factors; show that, for odd primes  $p$ , 1 and  $p^n - 1$  are the only square roots of 1 mod  $p^n$  (since  $\mathbf{Z}_{p^n}^\times$  is a cyclic group of even order); finally, find the square roots of 1 mod  $2^{\alpha_0}$ .
14. Multiply the polynomials  $2x^3 + 3x + 1$  and  $3x^2 + 3x + 1$ .

15. Show that polynomials satisfy properties A, C, and D. You may first want to notice that

$$\sum_i a_i x^i \cdot \sum_i b_i x^i = \sum_i \sum_{j+k=i} a_j b_k x^i$$

and then use properties A, C, and D of the coefficients' field.

16. Divide the  $\mathbf{Z}_5$  polynomial  $x^5+x^3+1$  by the  $\mathbf{Z}_5$  polynomial  $3x^2+2x+1$ .
17. In an analogous manner to §M.5 *Modular Arithmetic*, define polynomial addition and multiplication modulo a non-zero polynomial  $m(x)$ . Show that each definition makes sense in that it produces the same value (mod  $m(x)$ ) when you replace either of the operands with an equivalent (mod  $m(x)$ ) polynomial.
18. Show that the  $\mathbf{Z}_2$  polynomial  $m(x) = x^8+x^4+x^3+x+1$  is irreducible. Hint: First note that if a polynomial over a field is not irreducible, it has a nonconstant factor no larger than half its degree. Then show that  $x$  is a factor of a polynomial iff its constant term is 0, and that  $x+1$  is a factor of a  $\mathbf{Z}_2$  polynomial iff it has an even number of nonzero terms. If  $m(x)$  were not irreducible, it would have a nonconstant factor of degree 4 or less, so just thirty possible factors to try. Of those, you can quickly eliminate multiples of  $x$  and of  $x+1$ , since neither is a factor of  $m(x)$ , leaving just seven possible factors to try. And one of those is the square of one of the others, which you can see by noting that for  $\mathbf{Z}_2$  polynomials, where  $1+1=0$ ,  $(a(x)+b(x))^2 = (a(x))^2+(b(x))^2$ , so squaring a  $\mathbf{Z}_2$  polynomial can be accomplished by doubling the exponent of each term; so a  $\mathbf{Z}_2$  polynomial is a square iff its nonzero terms all have even exponents.
19. Compute the inverse of each nonzero  $\mathbf{Z}_2$  polynomial mod  $m(x)$ , where  $m(x) = x^8+x^4+x^3+x+1$ . Represent each polynomial as an octet, with the most significant bit being the coefficient of  $x^7$  and the least significant bit being the coefficient of  $x^0$ . Hint: Use the Euclidean algorithm, or raise each mod  $m(x)$  polynomial to the power  $2^8-2 = 254$ .
20. The Rijndael S-box is the composition of two permutations. The first is the inverse just described, but with 0 mapping to itself (since 0 doesn't have an inverse). This is clearly its own inverse. The second is multiplication by  $x^4+x^3+x^2+x+1 \bmod x^8+1$  followed by addition of  $x^6+x^5+x+1$ . What is its inverse?
21. Verify that the MixColumns polynomial  $c(x) = 03x^3+01x^2+01x+02$  and the InvMixColumns polynomial  $d(x) = 0Bx^3+0Dx^2+09x+0E$  are multiplicative inverses mod  $x^4+1$ , with all polynomials over  $\text{GF}(2^8)$  as represented in Rijndael. Extra credit: Find the inverse of  $c(x)$  by using the Euclidean algorithm.
22. Tabulate the Rijndael key expansion constants  $C_i = x^{i-1} \bmod m(x)$ , using the same representation as in Homework Problem 19.
23. Show that polynomials over a field form a unique factorization domain (in that every monic polynomial can be expressed uniquely as a product of powers of monic irreducible polynomials) by using the Euclidean algorithm to prove that if  $a$  divides  $bc$  and  $a$  is irreducible, then  $a$

divides  $b$  or  $a$  divides  $c$ . Big hint: If  $a$  doesn't divide  $b$ , then  $\gcd(a, b) = 1$ , so  $xa + yb = 1$ , so  $c = 1c = xac + ybc = a(xc + y(bc/a))$ .

24. Show that for every  $x$  and  $y$  in a field of characteristic  $p$ ,  $(x+y)^p = x^p + y^p$ . Hint: Use the binomial theorem, and note that the binomial coefficient  $\binom{p}{k}$  is a multiple of  $p$  for  $0 < k < p$ .
25. Show that, for any degree  $n$  monic irreducible polynomial  $\mathfrak{u}(x)$  over  $\mathbf{Z}_p$ , the  $\mathbf{Z}_p$  polynomials mod  $\mathfrak{u}(x)$  form a field of size  $q = p^n$ . Hint: It's easy to see that addition and multiplication satisfy properties A, C, D, and I, based on those properties for polynomials. Property N for addition is also easy, based on that property for  $\mathbf{Z}_p$ . Use the Euclidean algorithm to demonstrate property N for multiplication.
26. In this problem, we show that for each prime  $p$  and positive integer  $n$  there is exactly one field of order  $q = p^n$ . In the process, we derive a formula for the number of irreducible polynomials of order  $n$  over  $\mathbf{Z}_p$ . (Notation:  $a|b$  means  $a$  divides  $b$ .)
  - (A) Define the Möbius function  $\mu(n)$  for positive integer  $n$  to be 0 if  $n$  is divisible by the square of a prime, and otherwise  $-1$  or  $+1$  depending on whether it is divisible by an odd number or an even number of distinct primes, respectively. Show that if  $f(n)$  is any function defined on positive integers, and  $F(n) = \sum_{d|n} f(d)$ , then  $f(n) = \sum_{d|n} \mu(n/d)F(d)$ . Hint: Expand  $F(d)$  on the right hand side, and note that  $f(k)$  appears once for each  $d$  such that  $k|d$  but with coefficient  $\mu(n/d)$ . If  $m$  is a prime factor of  $n$ , and  $d$  is a divisor of  $n$  for which  $m$  doesn't divide  $n/d$ , then  $\mu(n/d) = -\mu(n/(d/m))$ .
  - (B) For prime  $p$ , and  $q = p^n$ , consider polynomials over  $\mathbf{Z}_p$ . Show that  $x^q - x$ , has no repeated factors. (Hint: We can define the **derivative** of a polynomial  $a(x) = \sum a_i x^i$  to be  $a'(x) = \sum (i+1)a_{i+1}x^i$ . Show that if  $c(x) = a(x)b(x)$ , then  $c'(x) = a'(x)b(x) + a(x)b'(x)$ . Conclude that if  $a(x)a(x)|c(x)$  then  $a(x)|c'(x)$ .) Show that every irreducible polynomial of degree  $n$  is a factor of  $x^q - x$ . (Hint: In Homework Problem 25, we showed how any irreducible polynomial  $\mathfrak{u}(x)$  of degree  $n$  generates a field of order  $q$ , where each element of that field is a polynomial modulo  $\mathfrak{u}(x)$ . We know that every element of that field must satisfy  $x^q - x$ . In particular, the polynomial  $x$  satisfies  $x^q - x$ , so  $x^q - x = 0 \pmod{\mathfrak{u}(x)}$ , i.e.  $\mathfrak{u}(x)|x^q - x$ .) Show that an irreducible polynomial of degree  $d$  is a factor of  $x^q - x$  iff  $d|n$ . (Hint: Show that  $\gcd(x^j - 1, x^k - 1) = x^{\gcd(j, k)} - 1$  and that  $\gcd(p^d - 1, p^n - 1) = p^{\gcd(d, n)} - 1$ .) Conclude that  $x^q - x$  is the product of all irreducible polynomials whose degree divides  $n$ . By counting roots, further conclude that every irreducible polynomial of degree  $d$  where  $d|n$  has  $d$  distinct roots in any field of order  $q$ .
  - (C) Define  $r_{p, n}$  to be the number of monic irreducible polynomials of degree  $n$  over  $\mathbf{Z}_p$ . Show that  $p^n = \sum_{d|n} d r_{p, d}$ , and conclude by (A) that  $n r_{p, n} = \sum_{d|n} \mu(n/d) p^d$ . Show that the right hand side is always positive and conclude that there exists at least one irreducible polynomial of each positive degree. Hint: One way to do this is to equate the degree of  $x^q - x$  with the sum of the degrees of its factors as shown in (B). Alternatively, by Homework Problem 23, every monic polynomial of degree  $n$  can be uniquely expressed as a product of powers of monic irreducible polynomials whose degrees times the respective powers add to  $n$ . There are  $p^n$

monic polynomials of degree  $n$ , so this can be expressed as  $\sum_n p^n x^n = \prod_i (\sum_j x^{ij})^{r_{p,i}}$ ; here the coefficient of  $x^n$  on the left hand side is the number of monic polynomials of degree  $n$ , and the expansion of the product on the right hand side chooses a power  $j$  for each of the  $r_{p,i}$  monic irreducible polynomials of degree  $i$ . The equation can be simplified to  $1 - px = \prod_i (1 - x^i)^{r_{p,i}}$ , where we've replaced the sums of geometric series and inverted both sides. Finally, take  $-\ln$  of both sides, noting that  $-\ln(1-z) = \sum_{k>0} z^k/k$ , and equate the coefficients of  $x^n$  on each side.

(D) Show that any two finite fields of order  $q$  are isomorphic, *i.e.*, there is a one-to-one mapping between elements of the two fields that preserves the field operators ( $+$  and  $\cdot$ ). Hint: In §M.7.2.2 *Representing a Field*, we showed that for any field  $F$  of order  $q = p^n$  we can represent each field element  $a$  as a linear combination  $a = \sum_{i<n} a_i g^i$ , where  $g$  is a generator of the multiplicative group  $F - \{0\}$ . With this representation, multiplication is determined by the degree- $n$  primitive polynomial  $\gamma(x)$  that  $g$  satisfies, while addition is just componentwise addition mod  $p$ . By (B), any field  $F'$  of order  $q$  must have an element  $g'$  that is a root of  $\gamma(x)$ . Map this element to  $g$ . This determines a one-to-one mapping between elements of  $F$  and  $F'$  where  $\sum_{i<n} a_i g^i \leftrightarrow \sum_{i<n} a_i g'^i$ , and this mapping clearly preserves the field operators.

27. How many degree  $n$  monic primitive polynomials over  $\mathbf{Z}_p$  are there? Hint: The roots of each such polynomial are all generators of  $\text{GF}(p^n)$ . How many generators does  $\text{GF}(p^n)$  have?

### matrices

28. Show that if  $\sum_{1 \leq i \leq m} a_i w_{ij} = b_j$  for  $1 \leq j \leq n$ , and  $\sum_{1 \leq i \leq n} b_i x_{ij} = c_j$ , then  $\sum_{1 \leq i \leq m} a_i y_{ij} = c_j$  where  $y_{ij} = \sum_{1 \leq k \leq n} w_{ik} x_{kj}$ . Hint: This is an application of distributivity.
29. Show that if  $Y = WX$ , then  $Y^T = X^T W^T$ , provided multiplication of elements is commutative. Hint: Just rearrange the indices.
30. Show that matrix addition is associative and commutative, *i.e.*,  $(A+B)+C = A+(B+C)$  and  $A+B = B+A$  for same-dimensioned matrices  $A$ ,  $B$  and  $C$ , as a direct consequence of those same properties of the matrix elements' ring. Show that matrix multiplication is associative and that it distributes over matrix addition, *i.e.*,  $A(BC) = (AB)C$  and  $A(B+D) = AB+AD$  and  $(A+E)B = AB + EB$  for all compatibly-dimensioned matrices  $A$ ,  $B$ ,  $C$ ,  $D$ ,  $E$ . Again, this is a straightforward application of those same properties of the matrix elements' ring.
31. Give an example of two  $2 \times 2$  matrices that don't multiplicatively commute, where each matrix element is either 0 or 1, and one of the products is the  $2 \times 2$  matrix with only 0 elements.
32. Show that for any  $m \times n$  matrix  $A$ ,  $\mathbf{I}_m A = A = A \mathbf{I}_n$ .
33. Show that if  $f$  is a function from  $1 \times m$  vectors to  $1 \times n$  vectors for which  $f(x+y) = f(x) + f(y)$  and  $f(ax) = af(x)$  for any  $a$  in the same ring with identity as the vector elements, then  $f$  is a linear transformation, *i.e.*, that applying  $f$  is equivalent to multiplying by an  $m \times n$  matrix. Notation:  $av$  is the vector obtained by multiplying  $a$  to each element of  $v$ . Hint: Consider what  $f$  does to

a vector whose elements are 0 except for one 1, then use the equations to see what  $f$  does to arbitrary vectors.

34. Show that if  $LM = I$  and  $MR = I$ , then  $L = R$ . Hint: Consider  $LMR$ .
35. Show that for any square invertible matrix  $M$ ,  $(M^{-1})^T = (M^T)^{-1}$ , or show a counterexample.
36. If you want to solve  $Ax = y$  or  $xA = y$  for  $x$ , in either case you want to divide  $y$  by  $A$ , *i.e.*, multiply by  $A^{-1}$ . For which of these equations is the answer  $x = A^{-1}y$  and for which is the answer  $x = yA^{-1}$ ?
37. Specify the elements of the matrices for transformations  $S_{rs}$ ,  $T_{m,r}$ , and  $A_{m,rs}$ , as defined in §M.12.2 *Matrix Inverses*.
38. In §M.12.2.1 *Gaussian Elimination*, show that if for some column  $c$  we can't find a row  $r \geq c$  with a nonzero element in column  $c$ , the matrix is not invertible. Hint: If we just skip the column and continue the algorithm, we end up with at least one row that's all zeroes (ignoring the  $n$  columns that were appended on the right where the inverse eventually appears).
39. Given a permutation  $\pi$ , show that the corresponding permutation matrix  $P$  has  $P_{ij} = \delta_{\pi(i), j}$ . Show that the permutation matrix for the inverse permutation  $\pi^{-1}$  is  $P^T$  and so  $P^T = P^{-1}$ . Show that  $\det(P) = \text{sgn}(\pi)$ . (See Homework Problem 8 for a permutation primer.)
40. A) Show that  $\det(A) = \det(A^T)$ . Hint: Use inverse permutations.  
 B) Show that the determinant of a triangular matrix is the product of its diagonal elements. Hint: Show that all the other terms in the determinant include at least one factor of 0.  
 C) Show that multiplying a row of a matrix by a constant ( $T_{m,r}$ ), *i.e.*, multiplying each element of the row by that constant, multiplies the matrix's determinant by that constant.  
 D) Show that swapping any two rows of a matrix ( $S_{rs}$ ,  $r \neq s$ ) negates its determinant. Hint: Swap is an odd permutation.  
 (E) Show that if any two rows of a matrix are identical, its determinant is zero. Nonhint: Why doesn't D imply this? Hint: If two rows are identical, each term in the determinant appears twice except for its sign.  
 F) Show that adding a multiple of a row to a different row in a matrix ( $A_{m,rs}$ ,  $r \neq s$ ) has no effect on the matrix's determinant. Hint: Use distributivity to break out the terms for the determinant of the original matrix, and notice what is left is the terms for the determinant of the matrix with the second row replaced by a multiple of the first.  
 G) Gaussian elimination. Assume the matrix elements come from a field. Show that you can use the above observations to slowly transform a matrix into a triangular matrix with the same determinant. At stage  $c$ , you will make sure that all rows below the diagonal will have only zeroes in column  $c$ : Swap row  $c$  with a later row if necessary to get a nonzero diagonal element  $A_{cc}$ , negate row  $c$  if you've done a swap, then subtract multiples of row  $c$  from each later row so that all those rows have zero in column  $c$ , but do nothing if all the later rows already have zero in column  $c$ . After the last stage, you will have a triangular matrix with the

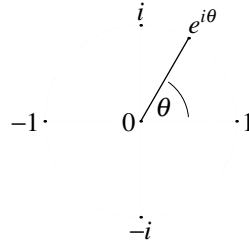


same determinant as the original.

If the determinant is nonzero, you can extend the above procedure to end up with an identity matrix: for each row  $c$ , divide each element by the diagonal element and then subtract the appropriate multiple of row  $c$  from each earlier row so that the only nonzero element in column  $c$  is the diagonal element (which is now 1). You have now performed a sequence of row operations that transformed the original matrix to the identity matrix. If you apply these same operations, in the same order, to the identity matrix, you will end up with the inverse of the original matrix. Why? Hint: Remember that matrix multiplication is associative, and each of the row operations can be represented by a matrix that multiplies on the left. What if you apply the sequence of row operations to a column vector? Hint: Consider the equation  $\mathbf{y} = \mathbf{A}\mathbf{x}$ .

41. Show that  $\text{adj}(\mathbf{M})\mathbf{M} = \mathbf{M}\text{adj}(\mathbf{M}) = \det(\mathbf{M})\mathbf{I}$ , i.e.,  $\mathbf{M}$  commutes with  $\text{adj}(\mathbf{M})$  and the product is a diagonal matrix each of whose diagonal elements is  $\det(\mathbf{M})$ . Hint: Each of the product's diagonal elements has exactly the terms that make up  $\det(\mathbf{M})$ , while each of the product's off-diagonal elements has exactly the terms that make up the determinant of  $\mathbf{M}$  with one of its rows or columns replaced by another.
42. A Vandermonde matrix  $\mathbf{V}$  is a matrix where  $V_{ij} = x_i^{j-1}$ . (Here we consider  $x^0 = 1$  for any  $x$ , so the first column is all 1s.) Show that, for a square Vandermonde matrix  $\mathbf{V}$ ,  $\det(\mathbf{V}) = \prod_{i < j} (x_j - x_i)$ . Hint: This is mostly a simple matter of matching up the terms of the resulting polynomials on the two sides of the equation; the hardest part is showing that the signs of the corresponding terms match. Conclude that if the  $x_i$ s are all distinct elements of a field,  $\mathbf{V}$  is invertible; and, conversely, if  $\mathbf{V}$  is invertible, the  $x_i$ s are all distinct. Vandermonde matrices find a use in §16.1 *Secret Sharing*.
43. Suppose  $\mathbf{u}$  and  $\mathbf{v}$  are vectors of coordinates in different linear coordinate systems, related by  $\mathbf{u} = \mathbf{T}\mathbf{v}$ . Let  $\mathbf{A}$  be a linear transformation in  $\mathbf{u}$ 's coordinate system. Show that the same transformation in  $\mathbf{v}$ 's coordinate system is  $\mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ .
44. For a matrix  $\mathbf{A}$  whose elements are complex numbers, the **conjugate transpose** of a matrix (also called the **Hermitian transpose**)  $\mathbf{A}^H$  is the transpose of the matrix but with each element replaced by its complex conjugate, i.e.,  $A_{mn}^H = \overline{A_{nm}}$ . Show that  $(\mathbf{AB})^H = \mathbf{B}^H\mathbf{A}^H$ .
45. A **unitary** matrix  $\mathbf{U}$  is a square matrix whose elements are complex numbers such that  $\mathbf{U}^H\mathbf{U} = \mathbf{I}$ , i.e., a matrix whose conjugate transpose is its inverse. Show that  $\mathbf{U}$  is unitary iff  $(\mathbf{U}\mathbf{v})^H(\mathbf{U}\mathbf{v}) = \mathbf{v}^H\mathbf{v}$  for all compatibly dimensioned vectors  $\mathbf{v}$  whose elements are complex numbers such that  $\mathbf{v}^H\mathbf{v} = 1$ .

46. Consider the  $N \times N$  Vandermonde matrix  $\mathbf{D}_{mn} = \exp(-2\pi i(m-1)(n-1)/N)$ , and its conjugate transpose  $\mathbf{D}_{mn}^H = \exp(2\pi i(m-1)(n-1)/N)$ . Show that  $\mathbf{D}^H \mathbf{D} = N\mathbf{I}$ . Hint:  $\exp(i\theta) = \cos\theta + i\sin\theta$



is the point on the 0-centered unit circle in the complex plane that is  $\theta$  counterclockwise from 1; any symmetrically arranged set of points on that circle sums to 0. Conclude that  $|\det(\mathbf{D})| = N^{N/2}$ .  $\mathbf{D}$  is called the **discrete Fourier transform (DFT)** matrix, because it converts a vector of equally-spaced samples into a vector of amplitudes of equally-spaced frequencies. It is often normalized by dividing each of its elements by  $\sqrt{N}$ ; show that then its inverse is its conjugate transpose, *i.e.*,  $\mathbf{D}/\sqrt{N}$  is unitary. Conclude from Homework Problem 42 that the product of the lengths of all chords between the vertices of a regular  $N$ -gon inscribed in a unit circle is  $N^{N/2}$ .