

Groups, Modular Arithmetic, and Cryptography

Jean Mark Gawron

Linguistics

San Diego State University

gawron@mail.sdsu.edu

<http://www.rohan.sdsu.edu/~gawron>

2004 July 24

Preface

The inspiration for this compilation of mathematical material comes from W. W. Sawyer's *Prelude to Mathematics*, an eclectic and beautiful tour of some of the most important ideas in mathematics, among them, matrices, groups, and non-Euclidean geometry. The book is unique both in the sophistication of its presentation and in the little background it assumes. Basically what is assumed is a reader willing to work a little.

The topics Sawyer discusses are chosen in part on aesthetic grounds. They all represent areas of what mathematicians call great mathematical beauty. To achieve that status a mathematical idea requires a certain amount of purity, distance from any of those crude particulars that infect mathematical applications. Yet in another sense what endows Sawyer's diverse ideas with their power is precisely that they have applications. It is the fact that these simple structural elements plug into so many diverse apparently unrelated areas that makes the mathematics beautiful. Non-Euclidean geometry is the root of a body of work that shows that geometries other than Euclidean are both coherent and conceptually fruitful. Both of Einstein's theories of relativity may be understood as such alternative geometries. Matrices have numerous mathematical lives. To mention just a random sample: they are used as operators in 3D geometry, as representations of Hidden Markov Models, as representations of permutations (discussed in Chapter 1).

It is thus a paradoxical combination of purity and applicability that makes the mathematics interesting. There is probably no clearer example of this than the applicability of modular arithmetic to public key cryptography.

Certainly before the advent of modern cryptography modular arithmetic could lay claim to being one of the purest (that is, most application-free) areas of mathematics. It was also, in its deep relationships to group and field theory, one of the most beautiful. With the advent of public key cryptography (particularly the RSA discussed here) applications of modular arithmetic and similar applications in polynomial fields, elliptic curves, and finite state automata, renewed interest on a number of mathematical fronts. Schneier (1996) provides an excellent survey.

[A little more explanation of pb key cryptography and its importance]

[3 factors in the growth of PK crypto: (a) processing power (old days: chips)
(b) dsitributed computing [Lotic: Iris: Notes] (c)internet]

Contents

1 Introduction	7
2 Groups	9
2.1 Introduction to Groups	9
2.1.1 Group Axioms	10
2.1.2 Exercises	14
2.2 Examples	15
2.2.1 Reals and Complex Numbers under Multiplication	15
2.2.2 Truth values under the operation “Exclusive Or”	16
2.2.3 Roots of $x^3 - 1 = 0$	16
2.2.4 Permutations	18
2.2.5 Exercises	26
2.3 Subgroups	27
2.3.1 Exercises	29
2.3.2 Morphisms	29
3 Modular Arithmetic	33
3.1 Modular Arithmetic	33
3.1.1 Remainders	33

3.1.2	Exercises	36
3.2	Grouphood	36
3.2.1	Grouphood of complete residue systems under addition mod n	37
3.2.2	Modular Arithmetic Groups with Multiplication	40
3.2.3	Exercises	42
3.3	Euclid's Algorithm and Euclid's Extended Algorithm	42
3.3.1	Euclid's Algorithm	43
3.3.2	Exercises	46
3.3.3	Euclid's Extended Algorithm	46
3.3.4	Exercises	50
3.4	Theorems about modular inverses	51
4	Group Properties of Multiplicative Groups	53
4.1	Grouphood of Z_n^*	53
4.1.1	Exercises	53
4.2	Subgroups and Cosets of Multiplicative Groups	54
4.3	LaGrange's Theorem	56
4.4	Fermat's Little Theorem and Euler's Theorem	58
4.4.1	Euler's Totient Function ϕ	58
4.4.2	Fermat's Little Theorem	58
4.4.3	Euler's Theorem	59
4.4.4	Exercises	62
5	Cryptography and Public Keys	63
5.1	Subject Matter of Cryptography	63
5.2	One-way functions	64
5.2.1	Fair play: A coin-tossing game problem	64

<i>CONTENTS</i>	5
5.2.2 A solution: A one-way function	65
5.2.3 The solution to several problems: Encryption	66
5.2.4 Verifiable key exchange: A practical almost one-way almost-function	68
5.2.5 Three true-to-life one-way functions	70
5.2.6 Verifiable key exchange: A one-way function protocol	71
5.2.7 Diffie-Hellman Protocol	72
5.2.8 Exercises	74
5.2.9 RSA protocol	74
5.2.10 Exercises	79
5.3 Practice versus Textbook	79
6 Appendix	83
6.1 Computing Euler's Totient Function	83
6.2 RSA Algorithm: What if m is not relatively prime to n	87

Chapter 1

Introduction

This compilation of mathematical material tries to assume very little. It is designed to be presented as part of a discrete mathematics course after some very introductory material has been presented. Thus, the elements of set theoretic notation and definitions are assumed, along with, for example, the notion of an equivalence relation.

The general mathematical background assumed is that of introductory college math. No knowledge of calculus is assumed but knowledge of high school algebra is essential, as is the ability to stare at equation in several unknowns without breaking into a cold sweat. This book is not intended to function as an independent introduction either to Group Theory or to modular arithmetic. The focus is on introducing some mathematical ideas in enough depth to understand their relationship to the cryptographic application. Thus in many cases methods of computing and manipulating the mathematical systems involved have been excluded. For example, the chapter on modular arithmetic does not cover the Chinese Remainder Theorem. The one case where computational method and mathematical idea seem to be inseparable, covered in some detail, is Euclid's Algorithm.

Each chapter develops applications of the ideas in the previous chapter. The material needs to be covered in roughly the order presented. The modular arithmetic chapter builds on and develops material in the groups chapter. The cryptography chapter presents a number of ideas that are independent of the mathematical developments, but ultimately the presentation of the Diffie-Hillman and RSA protocols rely on key ideas in modular arithmetic. The RSA protocol in particular is based on the concept of a multiplicative group in modular arithmetic, the set of the integers less than and relatively prime to some n under the operation of multiplication modulo n . Theorems in both group

theorem and modular arithmetic apply.

Chapter 2

Groups

In this chapter we discuss a fairly abstract mathematical notion, the notion of a group. A group is a very simple kind of mathematical system consisting of an operation and some set of objects. Our goal in this chapter is to learn just enough group theory to enhance our study of modular arithmetic in the next chapter, since the particular modular arithmetic systems that play a role in the cryptography chapter are groups. In particular we build up to the proof of a very important theorem known as Lagrange's Theorem in Chapter 4, which then leads to a very simple proof of Euler's Theorem, which directly underlies an important cryptographic application.

2.1 Introduction to Groups

A group (\mathbf{G}, \circ) begins with a set of objects \mathbf{G} that can be combined with some binary operation \circ . There are two main intuitions connected with the operation. First what we'll call **completeness** or **closure**. The operation is defined for every pair of objects in the set and always produces another member of the set. Second, there is an identity element, an element which, combined with any object, always returns that object. Finally, every member of the set has an **inverse element**, an element with which it can combine to return the identity element. As we'll see, this last property is connected with the idea that equations using this operation on unknowns can be solved with familiar techniques from basic algebra.

2.1.1 Group Axioms

Definition 2.1.1. Group

A group (\mathbf{G}, \circ) is a set \mathbf{G} together with an operation \circ satisfying the following basic properties:

1. **Closure Axiom.** The result of combining any two elements of \mathbf{G} with \circ always yields an element of \mathbf{G} .
2. **Associativity Axiom.** Order of combination does not matter in the following sense. For any 3 elements a, b , and c :

$$(a \circ b) \circ c = a \circ (b \circ c)$$

3. **Identity Element Axiom.** There is a distinguished element e such that for any a in \mathbf{G} :

$$e \circ a = a \circ e = a$$

4. **Inverse Axiom.** For any element a , there exists another elements called its **inverse**, which we will write a^{-1} , which is such that:

$$a^{-1} \circ a = a \circ a^{-1} = e$$

One way to understand each of these axioms is to contrast cases where they succeed and fail:

1. Closure: Failure. Positive integers and subtraction. $2-3$ is not a positive integer so the positive integers are not closed under subtraction. Success. Positive integers under addition. The sum of any two positive integers is a positive integer, so the positive integers are closed under addition.
2. Associativity: Subtraction is not associative.

$$\begin{aligned} (5 - 3) - 2 &= 0 \\ 5 - (3 - 2) &= 4 \end{aligned}$$

Addition is associative:

$$\begin{aligned} (5 + 3) + 2 &= 10 \\ 5 + (3 + 2) &= 10 \end{aligned}$$

3. Identity element: The set of positive integers under addition includes no identity element, since 0 is the identity element for addition. For just that reason, the set of positive integers plus 0 does satisfy the identity element axiom.

4. Inverse. The set of positive integers plus 0 under addition does not include inverses, since the inverse of a positive integer under addition is always a negative integer. The set of **all** non zero integers does satisfy the inverse axiom.

Much of our focus in this book is on groups with a finite number of members, for which the following notion is useful:

Definition 2.1.2. *Order of a Group*

*The number of elements in a finite group (\mathbf{G}, \circ) is called the **order** of \mathbf{G} and is denoted by $|\mathbf{G}|$.*

Example: Integers under addition (written $(\mathbf{Z}, +)$) The integers \mathbf{Z} (positive and negative integers and 0) are a group under addition. We write $(\mathbf{Z}, +)$

1. Closure: The sum of two integers is an integer
2. Associativity: Addition is asociative

$$(a + b) + c = a + (b + c)$$

3. Identity: 0 is the identity element.
4. The additive inverse of any integer i is $-i$.

Once an operation is defined, the trick to defining a group is often to choose the right set. Consider multiplication. Multiplication is associative:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

But note that the integers under multiplication are not a group. Closure, associativity and identity are satisfied, with 1 being the identity element. But the inverse of an integer under multiplication is generally not an integer.

A natural method for defining a group, a method we shall use throughout, is to begin by defining an operation and then choosing some particular set of appropriate objects which form a group *under that operation*. Often the set needs to be chosen with some care to satisfy all the clauses of the group definition. Thus the set of integers is a group under addition but the set of positive integers is not.

Consider the case of multiplication again. The set of integers does not give a group under the operation of multiplication because the inverses for that operation are not included in that set. If however we consider the set of **rational**

numbers, written \mathbf{Q} (all numbers expressible as a ratio between natural numbers), then fractions and whole numbers alike are included ($\frac{5}{1}$ is included along with its inverse $\frac{1}{5}$) But this is not quite good enough because there is still one element missing an inverse, 0 (what do you multiply 0 by to get 1?) So the final trick to defining a group under multiplication is to exclude 0:

The set of rationals \mathbf{Q} excluding 0 is a group under multiplication (written $(\mathbf{Q} - \{0\}, *)$).

1. Closure: The product of two rationals is a rational
2. Associativity: Multiplication is associative.
3. Identity: 1 is the identity element.
4. The multiplicative inverse of a rational number $\frac{i}{j}$, $i, j \in \mathbf{Z}$ is a rational number, namely $\frac{j}{i}$.

The set of positive rationals (written \mathbf{P}) is therefore also a group, since the product of two positive numbers is positive and since the identity element and inverses are included.

Our first theorem about groups gets at the notion of why groups have solvable equations.

Theorem 2.1.1. *Cancellation*

- (a) If $a \circ b = a \circ c$ then $b = c$ *Left Cancellation*
 (b) If $b \circ a = c \circ a$ then $b = c$ *Right Cancellation*

Proof:

We show the proof for case (a). Case (b) is symmetric. Assume:

$$a \circ b = a \circ c$$

From this we have:

- (1) $a^{-1} \circ a \circ b = a^{-1} \circ a \circ c$ *Inverse Axiom*
 (2) $(a^{-1} \circ a) \circ b = (a^{-1} \circ a) \circ c$ *Associative Axiom*
 (3) $e \circ b = e \circ c$ *Inverse Axiom*
 (4) $b = c$ *Identity Axiom*

Q.E.D.

What Theorem 2.1.1 essentially tells us is that given an a and the result of combining a with any other element b , b is recoverable (the operation can be *cancelled*).

Theorem 2.1.1 depends crucially on the existence of inverses to go through. This property is also crucially what is required in solving an algebraic equation. Here we step through a simple bit of algebra a little slower than usual:

$$\begin{array}{ll} \text{To solve: } 5 \cdot x = 20 & \\ (1) \quad 5^{-1} \cdot 5 \cdot x = 5^{-1} \cdot 20 & \text{Inverse Axiom} \\ (2) \quad (5^{-1} \cdot 5) \cdot x = (5^{-1} \cdot 20) & \text{Associative Axiom} \\ (3) \quad 1 \cdot x = 4 & \text{Inverse Axiom} \\ (4) \quad x = 4 & \text{Identity Axiom} \end{array}$$

If we know the result of multiplying 5 times x , x is recoverable and unique.

Note that grouphood guarantees us that *all* such equations have unique solutions as long we stick to group members on both sides. This goes along with the idea that the set we are performing operations on has to be “big enough” to be a group. If it is big enough to be a group it is big enough to contain all the solutions to such equations. The set of integers wasn’t big enough to be a group under multiplication and indeed, it isn’t big enough to include all the solutions to equations involving multiplication, even multiplication with integers. For example, the solution to

$$2x = 5$$

isn’t an integer.

Definition 2.1.3. *Iterated Group Operations*

It follows from the axioms that

$$a \circ a \in \mathbf{G}$$

It also follows that

$$(a \circ a) \circ a = a \circ (a \circ a) \in \mathbf{G}$$

We write a^2 for

$$a \circ a$$

and a^3 for

$$a \circ a \circ a$$

and in general we write a^i for i repeated applications of \circ to a .

Note that for this notation to make sense i must be an integer. It is important to remember that this notation doesn’t mean that the integer i is a member of

the group, or that the operation between a and i is a group operation. As we'll see when we look at some examples, there are lots of groups that have no integers as members, but even for those groups it makes sense to write a^i for i repeated applications of the group operation \circ to a .

Note also that every so often this notation will conflict with ordinary arithmetic usage. So within $(\mathbf{Z}, +)$

$$\begin{aligned}1^2 &= 1 + 1 = 2 \\2^3 &= 2 + 2 + 2 = 6\end{aligned}$$

Definition 2.1.4. *Cyclic Group, Group Generator . A group \mathbf{G} is cyclic if there exists $a \in \mathbf{G}$ such that for any $b \in \mathbf{G}$, there is an integer $i \geq 0$ such that*

$$a^i = b$$

Thus every element of \mathbf{G} is some power of a . Element a is called the generator of \mathbf{G} , which we write $\mathbf{G} = \langle a \rangle$

Example: -1 in $(\{1, -1\}, *)$. Consider the operation of multiplication over the set $\{-1, 1\}$. This is a group with 1 as the identity element. -1 is its own inverse and it is also the generator of the group since

$$\begin{aligned}(-1)^2 &= 1 \\(-1)^3 &= -1\end{aligned}$$

Note that 1 is not a generator because no amount of multiplying 1 by itself yields -1 .

In $(\mathbf{Z}, +)$ 1 generates the entire set of positive integers:

$$\begin{aligned}1^1 &= 1 \\1^2 &= 2 \\&\vdots \\1^n &= n\end{aligned}$$

but it does not generate the entire group because it does not generate the negative integers.

We'll see some more interesting cases of group generators next chapter.

2.1.2 Exercises

1. Consider the potential group (\mathbf{G}, \circ) where

$$\mathbf{G} = \{a, b, c, d, e, f\}$$

and \circ is defined by the following table (note e in this case is not necessarily the identity element):

(\mathbf{G}, \circ)

	a	b	c	d	e	f
a	a	b	c	d	e	f
b	b	d	f	a	c	e
c	c	f	b	e	a	d
d	d	a	e	b	f	c
e	e	c	a	f	d	b
f	f	e	d	c	b	a

- Refute or verify the grouphood of (\mathbf{G}, \circ) . You may assume the operation is associative.
- Compute e^5 .
- Refute or verify the grouphood of (\mathbf{G}', \circ) where

$$\mathbf{G}' = \{a, b, d\}$$

- Axiom 3 guarantees there's an identity element that is the identity element for every element of the group. But it doesn't say there is only one such element. Prove that the identity element is unique. That is, prove that if (\mathbf{G}, \circ) is a group then if $a, b \in \mathbf{G}$ and $a \circ b = a$, then $b = e$.
- Show that

$$(a^{-1})^{-1} = a$$

That is, show that the inverse of the inverse of a is a again.

- Show that

$$\text{If } a^{-1} = b^{-1} \text{ then } a = b$$

2.2 Examples

2.2.1 Reals and Complex Numbers under Multiplication

We needed to expand our view from the natural numbers to all integers to form a group under addition, from integers to all rationals excluding 0 to get a group under multiplication.

If we continue to expand our view to real and complex numbers, \mathfrak{R} and \mathbb{C} , still excluding 0, we still have groups under multiplication.

2.2.2 Truth values under the operation “Exclusive Or”

Consider the idea of “exclusive or”:

$$p \oplus q$$

is true if either p or q is true but not if both are true.

We can represent the exact definition in the form of what is called a **truth table**:

Truth Table for Exclusive Or (2.1)

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

This is the way the logician would represent the definition of \oplus . This way of representing the information will be important later on in the course. To an algebraist all this talk of ps and qs and things that *bear* truth values is unnecessary clutter. The meat of the matter may be expressed in the following table:

Algebraist’s Exclusive Or (2.2)

	T	F
T	F	T
F	T	F

Note that (2.1) and (2.2) define the same operation on truth values.

Then $(\{\mathbf{T}, \mathbf{F}\}, \oplus)$ is a group:

1. Closure: The exclusive or of two truth values is a truth value.
2. Associativity: See exercise 1.
3. Identity: See exercise 1.
4. Inverse: See exercise 1

2.2.3 Roots of $x^3 - 1 = 0$

The roots of the equation

$$x^3 - 1 = 0$$

form a group under multiplication. There are 3.

One of these roots is 1, which will be the identity element and therefore its own inverse. The other two are complex numbers.

The equation can be re-written:

$$x^3 - 1 = (x - 1)(x^2 + x + 1) = 0$$

Thus the other two roots are the roots of

$$x^2 + x + 1 = 0 \tag{2.3}$$

It turns out if you plug this equations into the formula that governs such things (the quadratic formula) these two roots **a** and **b** are such that:

$$\mathbf{ab} = 1$$

which means they are each other's inverses.

To see this, the formula for solving a quadratic equation of the form:

$$ax^2 + bx + c = 0$$

is:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.4}$$

The parameters a, b , and c in (2.3) are all equal to 1. Plugging those values into (2.4), we get:

$$x = \frac{-1 \pm \sqrt{1 - 4}}{2} \tag{2.5}$$

$$= \frac{-1 \pm \sqrt{-3}}{2} \tag{2.6}$$

$$= \frac{-1 \pm \sqrt{3}i}{2} \tag{2.7}$$

$$\tag{2.8}$$

Multiplying these two numbers together we have:

$$\frac{-1 + \sqrt{3}i}{2} \cdot \frac{-1 - \sqrt{3}i}{2} = \frac{1 - 3i^2}{4} \tag{2.9}$$

$$= \frac{1 - (-3)}{4} \tag{2.10}$$

$$= \frac{1 + 3}{4} \tag{2.11}$$

$$= 1 \tag{2.12}$$

The fact that these numbers are also cube roots of 1 follows from (2.12) and from the very peculiar fact that these numbers are each other's square roots as well! That is:

$$\left(\frac{-1 + \sqrt{3}i}{2}\right)^2 = \frac{-1 - \sqrt{3}i}{2}$$

and

$$\left(\frac{-1 - \sqrt{3}i}{2}\right)^2 = \frac{-1 + \sqrt{3}i}{2}$$

It turns out that there is only one way for things to work out in a 3-element group. If a and b are the two non-identity elements then it has to turn out that

$$\begin{aligned} a^2 &= b \\ b^2 &= a \end{aligned}$$

and it has to turn out that a and b are each other's inverses. Therefore it has to be the case that:

$$a^3 = b^3 = e$$

The reader should verify that:

	a	b	e
a	b	e	a
b	e	a	b
e	a	b	e

is the only possible definition for the operation \circ if $(\{a, b, c\}, \circ)$ is a group and the set really has 3 distinct members. In particular consider how the following operation table is excluded:

	a	b	e
a	e	?	a
b	?	e	b
e	a	b	e

This is the case where a and b are their own inverses. The '?'s can be replaced with any of the 3 elements. In all cases, the resulting system will not be a 3-member group. HINT: Consider what the Cancellation theorem has to say about how the '?' s must be filled in.

2.2.4 Permutations

A permutation is a way of mapping a set onto itself. A good example of a permutation is a cipher, a code that maps each letter of the alphabet onto a

different letter of the alphabet: For example this simple cipher maps each letter onto the next

$$\begin{bmatrix} A \rightarrow B \\ B \rightarrow C \\ D \rightarrow E \\ \vdots \\ Y \rightarrow Z \\ Z \rightarrow A \end{bmatrix}$$

This cipher would transform the message, "hello world" as follows:

$$\begin{array}{ccccccccc} H & E & L & L & O & W & O & R & L & D \\ & & \Downarrow & & & & \Downarrow & & & \\ I & F & M & M & P & X & P & S & M & E \end{array}$$

This kind of cipher, where each letter is shifted a constant amount, is called a *shift cipher*.

A famous example of a shift cipher is Caesar's cipher, so-called because Julius Caesar is supposed to have used it for military dispatches. This cipher shifts each letter 3 letters:

$$\begin{bmatrix} A \rightarrow D \\ B \rightarrow E \\ D \rightarrow F \\ \vdots \\ Y \rightarrow B \\ Z \rightarrow C \end{bmatrix}$$

Readers interested in keeping their communications secure are counseled not to use a shift cipher. After all there are only 26 of them and it wouldn't take a computer very long to figure out which one encodes any particular message. In fact, though it might take a few cycles longer, no cipher is secure.

Nevertheless to help illustrate some ideas in group theory we shall take a somewhat closer look at ciphers. To simplify the discussion, the first thing we need to do is map all the elements of the set being permuted to natural numbers. This is called choosing an indexing. Since the letters of the alphabet have an order there's a natural indexing for them, A=0,B=1,...Z=25. So Caesar's cipher

can be represented as a permutation on natural numbers.

$$\begin{bmatrix} 0 & \rightarrow & 3 \\ 1 & \rightarrow & 4 \\ 2 & \rightarrow & 5 \\ & & \vdots \\ 24 & \rightarrow & 1 \\ 25 & \rightarrow & 2 \end{bmatrix}$$

Permutations can of course be represented as functions. The above shift can be represented as a function σ that adds 3 to all the integers from 0 through 22 and maps 23 to 0, 24 to 1 and 25 to 2. For example we could write:

$$\begin{bmatrix} \sigma(0) & = & 3 \\ \sigma(1) & = & 4 \\ \sigma(2) & = & 5 \\ & & \vdots \\ \sigma(24) & = & 1 \\ \sigma(25) & = & 2 \end{bmatrix}$$

For a function to represent a permutation it must be 1-1. That is, no two elements can be mapped to the same thing.

A useful consequence of representing permutations as functions is that we can represent the **composition of permutations** (the result of applying one permutation after applying another) as a **composition of functions**. Thus if we apply a shift permutation that shifts letters one position to a permutation that shifts them 3, we should get a permutation that shifts letters 4 positions. And this is indeed the composition of the two functions representing those shifts. That is, let τ be the shift-1 permutation:

$$\begin{bmatrix} \tau(0) & = & 1 \\ \tau(1) & = & 2 \\ \tau(2) & = & 3 \\ & & \vdots \\ \tau(24) & = & 25 \\ \tau(25) & = & 0 \end{bmatrix}$$

Then the composition of the σ and τ permutations is given by $\tau \circ \sigma$:

$$\begin{bmatrix} \tau \circ \sigma(0) & = & 4 \\ \tau \circ \sigma(1) & = & 5 \\ \tau \circ \sigma(2) & = & 6 \\ & & \vdots \\ \tau \circ \sigma(24) & = & 2 \\ \tau \circ \sigma(25) & = & 3 \end{bmatrix}$$

One can also represent a permutation as a matrix, which is what we will do here. For an m by n matrix A , we write a_{ij} for the element in the i th row and j th column of matrix A .

$$\begin{bmatrix} & \text{col 1} & \text{col2} & \dots & \text{col j} & \dots & \text{col n} \\ \text{row 1} & 1 & 0 & \dots & 0 & \dots & 1 \\ \text{row 2} & 0 & 1 & \dots & 0 & \dots & 0 \\ & & & & & & \\ \text{row i} & 0 & 0 & \vdots & a_{ij} & \dots & 0 \\ & & & & & & \\ \text{row m} & 0 & 0 & \dots & 0 & \dots & 0 \end{bmatrix}$$

Then the above function σ can be turned into a matrix with the following convention:

$$\begin{aligned} a_{ij} &= 1 \text{ only if } \sigma(i) = j \\ \text{Otherwise, } a_{ij} &= 0 \end{aligned}$$

For ease of display let's use for our first example an alphabet with only five letters to permute. Thus our σ is entirely given by:

$$\begin{bmatrix} \sigma(0) = 3 \\ \sigma(1) = 4 \\ \sigma(2) = 0 \\ \sigma(3) = 1 \\ \sigma(4) = 2 \end{bmatrix}$$

This could be represented by the following matrix:

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (2.13)$$

So reading the 0 th (top) row: The 0 th row tells us what 0 gets mapped to. There is a 1 in the 3 rd column (counting the leftmost column as the 0 th). This means 0 gets mapped to 3 . Reading the next row, which tells us what 1 gets mapped to: 1 gets mapped to 4 . And so on. Note that the fact this is a shift cipher (the symbols are shifted by a constant amount) is reflected by the fact that on successive rows the 1 s are shifted 1 column over (putting them on a diagonal).

As a second example, the shift that shifts letters just one value (call it \mathbf{S}) is:

$$\mathbf{S} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.14)$$

The shift that goes to opposite extreme and shifts everything 4 letters (call it \mathbf{X}) is:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.15)$$

And the shift that leaves everything alone is:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.16)$$

Note that in a properly drawn permutation matrix each row and each column should have exactly one 1. That is, each input symbol (row) should get mapped to some output symbol and to only one output symbol. And each output symbol (column) should come from some input symbol and only one input symbol.

We don't have a group until we define an operation. That operation will be **matrix multiplication**, denoted by \cdot . The result of multiplying two matrixes A and B will be a third matrix C given by the following rule:

$$c_{ij} = \sum_{k=1}^n a_{ik} \cdot b_{kj}$$

The \cdot operation inside the summation is ordinary multiplication, so the assumption is that the elements in the matrix are all integers. If you look carefully at the way the definition works the assumption is also that A has exactly the same number of columns as B has rows, namely n . So the definition works for an A that is m by n and a B that is n by l . And the result is a matrix that is m by l .

The intuition of the definition, to the extent there is one at this point, can be stated by the following procedure. Consider A, an m by n matrix, and B, an n by l matrix, We fill cell c_{ij} of C by taking the i th row of A and combining

it with the j th column of \mathbf{B} . The i th row of \mathbf{A} can be thought of as an 1 by n matrix and the j th column of \mathbf{B} as an n by 1 matrix:

$$\begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{i(n-1)} & a_{in} \end{bmatrix} \cdot \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{(n-1)j} \\ b_{nj} \end{bmatrix}$$

Now multiply each cell of the \mathbf{A} row with the corresponding cell of the \mathbf{B} column. This gives us n products. The value of c_{ij} is the sum of these n products.

For example,

$$\begin{aligned} & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \\ & = (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) + (0 \cdot 0) \\ & = 0 + 0 + 0 + 0 + 0 \\ & = 0 \end{aligned}$$

On the other hand,

$$\begin{aligned} & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \\ & = (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 1) + (0 \cdot 0) \\ & = 0 + 0 + 0 + 1 + 0 \\ & = 1 \end{aligned}$$

We can represent the result of permuting according to a permutation (or encoding, if we are talking about ciphers) with matrix multiplication. We illustrate this with permutation \mathbf{B} in (2.13). First, we need some data to permute. An input character will be represented by a 1 by 5 matrix (remember our example input alphabet only has 5 characters). For example, let us set matrix \mathbf{A} to the character 3:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

To compute the encoding of \mathbf{A} by the shift cipher \mathbf{B} in (2.13) we simply matrix

multiply **A** by **B**. That is, we compute $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$:

$$\begin{array}{ccc} \mathbf{A} & \cdot & \mathbf{B} & = & \mathbf{C} & (2.17) \\ [0\ 0\ 0\ 1\ 0] & \cdot & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} & = & [0\ 1\ 0\ 0\ 0] \end{array}$$

This says character 3 gets mapped to character 1 by the shift-3 permutation. Figure 2.1 shows the step-by-step multiplication with the relevant column in bold:

The elegant property of the matrix approach to permutations is that we may also represent compositions of permutations by matrix multiplication; and this is what leads to a group structure. The reader may verify that **S** (the matrix for shifting one letter in 2.14) multiplied by our shift-three matrix **B** gives **X**, the shift-4 matrix (given in 2.15):

$$\begin{array}{ccc} \mathbf{S} & \cdot & \mathbf{B} & = & \mathbf{X} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} & \cdot & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} & = & \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\ & & & & (2.18) \end{array}$$

In the exercises, we informally verify that the set of n by n permutation matrices form a group under matrix multiplication.

This somewhat extended example of a group is intended to bring home the point that the elements of a group do not need to be numbers and the operation does not need to be a familiar arithmetic operation. In this case the elements of the permutation group are matrices and the operation is matrix multiplication. Note that the operation is not commutative. That is, in general

$$\mathbf{X} \cdot \mathbf{Y} \neq \mathbf{Y} \cdot \mathbf{X}$$

The reader should verify this with an example. For instance:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \circ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The notion of a permutation is very important in mathematics. Mathematically, a permutation is any function f from a set X to X which is 1-to-1 and onto. Because each row in a group Cayley table contains all the members of G with no repetitions, each row represents a permutation of G . For example, here

$$\begin{aligned}
[00010] \cdot \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix} &= [0 \quad] \\
\mathbf{C}_{00} &= (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) + (0 \cdot 0) \\
\mathbf{C}_{00} &= 0 \\
[00010] \cdot \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix} &= [01 \quad] \\
\mathbf{C}_{01} &= (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 1) + (0 \cdot 0) \\
\mathbf{C}_{01} &= 1 \\
[00010] \cdot \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix} &= [010 \quad] \\
\mathbf{C}_{02} &= (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 1) \\
\mathbf{C}_{02} &= 0 \\
[00010] \cdot \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix} &= [0100 \quad] \\
\mathbf{C}_{03} &= (0 \cdot 1) + (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0) \\
\mathbf{C}_{03} &= 0 \\
[00010] \cdot \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix} &= [01000 \quad] \\
\mathbf{C}_{04} &= (0 \cdot 0) + (0 \cdot 1) + (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 0) \\
\mathbf{C}_{04} &= 0
\end{aligned}$$

Figure 2.1: Encoding of the input $[00010] = 3$

is the b row in a group with 4 elements. This row maps a to c , b to e (b is its own inverse), and so on:

	a	b	c	e
	...			
b	c	e	a	b
	...			

What we have argued in this section, using the special case of 5 by 5 matrices in our examples, is that the set of permutations of a set is a group under composition. We have not drawn the Cayley tables for our permutation groups because they get rather large. There are 25 permutations of a set of 5 elements, so the Cayley table has 625 cells (25 by 25).

2.2.5 Exercises

1. Prove the associativity, identity, and inverse axioms for $(\{\mathbf{T}, \mathbf{F}\}, \oplus)$.
2. Here are the truth tables for some more operations on truth values. Which form the basis of a group?

(a) And (\wedge)

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

(b) Or (\vee)

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

(c) Implies (\rightarrow)

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

(d) If-and-only-if (Equivalence, \leftrightarrow)

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

3. Compute

$$\mathbf{S} \cdot \mathbf{X}$$

where \mathbf{S} and \mathbf{X} are as in example (2.18).

4. Verify that the set of n by n permutation matrices form a group under matrix multiplication informally by looking at the case of 5 by 5 matrices, as in the discussion and doing the following:

- (a) Find the identity element.
- (b) Find inverses for \mathbf{I} , \mathbf{S} , \mathbf{X} , and \mathbf{B} , where these are the matrices in (2.16) and (2.18). What is the generalization, stated informally?
- (c) Proving associativity for matrix multiplication is no mean feat. But construct an informal argument. Hint: You may appeal to the connection between multiplication of permutation matrices and function composition.

2.3 Subgroups

Definition 2.3.1. *Subgroup A subgroup of a group (\mathbf{G}, \circ) is a nonempty subset \mathbf{H} of \mathbf{G} which is itself a group under \circ . We write $\mathbf{H} \sqsubseteq \mathbf{G}$ to denote that \mathbf{H} is a subgroup of \mathbf{G} .*

The set of even integers forms a subgroup of the set of integers under addition. The sum of two even numbers is even; the inverses of even numbers are even, and the identity element, 0, is even. Notice that each subgroup must include the identity element in order to be a group. The set of odd integers does not form a subgroup under addition because closure fails (the sum of odd integers is even); note also the identity element 0, is not odd. In general, the set of numbers divisible by any integer n will form a subgroup of the integers under addition. The reader should verify this.

The set of positive rationals \mathbf{P} is a subgroup of the set of rationals $\mathbf{Q} - \{0\}$ under the operation of multiplication.

As a final example, consider a group (\mathbf{G}, \circ) and a member a , and consider the smallest integer i such that:

$$a^i = e$$

where e is the identity element.

Definition 2.3.2. *Order of an element of a group*

Let i be the smallest integer such that:

$$a^i = e$$

where a is any element of a group \mathbf{G} of which e is the identity element. If i exists we call i the order of a . If not, we say a has infinite order.

Theorem 2.3.1. *The subgroup generated by an element*

For any a with finite order in group \mathbf{G} , the set

$$\mathbf{H} = \{x \mid x = a^j \text{ for some integer } j\}$$

is a subgroup of \mathbf{G} under \circ . Let i be the order of a . Note that

$$\begin{aligned} a^i &= e = a^0 \\ a^{i+1} &= a = a^1 \\ a^{i+2} &= a^2 \\ &\vdots \\ a^{i+n} &= a^n \end{aligned}$$

and for all higher values of j , we cycle through previously encountered values. Clearly closure is satisfied. Every power of a combined with another power of a gives a power of a . Note also that:

$$a^{-1} = a^{i-1} \text{ because } a \circ a^{i-1} = a^i = e$$

*We can similarly find an inverse for every other element of \mathbf{H} . Note that \mathbf{H} satisfies the definition of a cyclic group given in Definition 2.1.4. Element a is its **generator**.*

Example 2.3.1. *The subgroup generated by T in the exclusive-or subgroup*

As an example, for the exclusive-or group given in Section 2.2.2, the order of T was 2, so T in fact generates the whole group, and the order of F is 1.

Example 2.3.2. *The subgroup generated by \mathbf{a} and \mathbf{b} in the $x^3 - 1$ group*

For the group consisting of roots of $x^3 - 1$ the order of \mathbf{a} and \mathbf{b} is 3, so both are generators of the entire group.

What Theorem 2.3.1 illustrates is a sure fire way of generating subgroups: Just collect the set of all the powers of some element a of finite order.

It turns out that if we restrict our attention to finite groups, all elements generate a subgroup. There are no elements of infinite order in finite groups. We leave the proof as an exercise, explained in more detail in Section 2.3.1.

2.3.1 Exercises

1. Show that if \mathbf{H} , a subgroup of \mathbf{G} , has only one element, a , then a must be the identity element of \mathbf{G} .
2. Consider \mathbf{G} , a finite group. Show that \mathbf{G} has no elements of infinite order.

HINT: \mathbf{G} is finite. Therefore, for any a , there exist integers i, j , $0 < i < j$, such that:

$$a^i = a^j \quad (2.19)$$

In other words, as we keep raising a to higher and higher powers, some results must be repeated, because there are only a finite number of possibilities to cycle through. To show there are no elements of infinite order it will suffice to produce some positive x such that $a^x = e$.

3. Assume the result of the previous exercise has been proven. Now show that in a group the inverse of any element a is always a positive power of a . That is, show that there is always some $k > 0$ such that:

$$a^k = a^{-1}$$

Don't think too hard about this. Given the result of the previous exercise, this does follow pretty straightforwardly.

2.3.2 Morphisms

A *morphism of groups* is a function from the members of one group \mathbf{G} to the members of another \mathbf{G}' that preserves the structure of the group operation.

$$f : G \rightarrow G'$$

Let \circ be the operation in \mathbf{G} and \circ' the operation in \mathbf{G}' . Then for all $a, b \in \mathbf{G}$:

$$f(a \circ b) = f(a) \circ' f(b)$$

Recall that the set of positive rationals \mathbf{P} is a group under multiplication and the set of rational numbers \mathbf{Q} is a group under addition. Then the log function

is a morphism of groups from P to Q because:

$$\log(x * y) = \log x + \log y$$

For example:

$$\log 1 = 0 \quad (2.20)$$

$$\log 10 = 1 \quad (2.21)$$

$$\log(10 * 1) = 1 + 0 = 1 \quad (2.22)$$

$$\log 1000 = 3 \quad (2.23)$$

$$\log(10 * 1000) = 1 + 3 = 4 \quad (2.24)$$

Note in particular the following properties:

$$(a) \log(1) = 0 \quad (2.25)$$

$$(b) \log\left(\frac{1}{1000}\right) = -3 \quad (2.26)$$

$$(2.27)$$

The identity element of multiplication is mapped onto the identity element of addition. The inverse of 1000 for multiplication is mapped onto the inverse of $\log 1000$ for addition. These properties are true of all morphisms of groups:

$$(a) f(e) = e' \quad (2.28)$$

$$(b) f(x^{-1}) = (f(x))^{-1} \quad (2.29)$$

$$(2.30)$$

The identity element in G is mapped to the identity element in G' and the inverse of x is mapped to the inverse of $f(x)$. These properties follow from the definition of a morphism. Proof left as an exercise. Given these two facts, it is not hard to show that the image of a morphism to G' always identifies a subgroup of G' (proof again left as an exercise). If f is onto, that subgroup is G' itself.

Theorem 2.3.2. *Morphism from Z to G for any fixed element*

For any fixed element g of a group G , the mapping

$$\phi : n \mapsto g^n :$$

is a morphism from Z to a subgroup of G . As noted in the previous section the powers of any element g form a subgroup with m elements where m is the order of g . Integers higher than m repeat previous values of ϕ , so ϕ is not 1-to-1.

For this to make sense, we need to define negative powers and 0:

$$g^0 = e$$

$$g^{-n} = (g^n)^{-1}$$

A rigorous proof is omitted here but note that, as desired:

$$\phi(m+n) = g^{m+n} = g^m \circ g^n$$

This follows directly from the definition of g^n .

Chapter 3

Modular Arithmetic

3.1 Modular Arithmetic

3.1.1 Remainders

Consider the set of integers that give a remainder of 4 when divided by 7:

$$\{x \mid x \text{ divided by } 7 \text{ gives a remainder of } 4\}$$

Examples:

$$4, 11, 18, 25, 32, 39$$

Notice if we arrange these numbers in ascending order they form a series. You add 7 to the last element to get the next. Now consider the set of numbers that give a remainder of 3 when divided by 7.

$$\{x \mid \exists q \in \mathbb{N} [(q \cdot 7) + 3 = x]\}$$

Examples:

$$3, 10, 17, 24, 31, 38, \dots$$

Notice we again have a series. And again you add 7 to the last element to get the next.

We say that the first set the set of numbers is *congruent to 4 mod 7* and that the second the set of numbers *congruent to 3 mod 7*. We use the symbol \equiv to express the congruence relation. For example we write:

$$4 \equiv 11 \equiv 32 \pmod{7}$$

The congruence relation establishes the equivalence of certain numbers for certain purposes, namely **modular arithmetic**. Since all the congruent numbers are equivalent we generally make do with just one of them, usually the smallest positive number, in this case 4. Thus 4 can stand for any of the numbers that give a remainder of 4 when divided by 7. Computationally, we can think of ourselves as *converting a number to its modulus representation* by dividing by the modulus, throwing away the quotient, and keeping the remainder:

$$\begin{aligned} 39 \div 7 &= 5 \text{ remainder } 4 \\ 39 &\equiv 4 \pmod{7} \end{aligned}$$

All we need is the set of remainders when dividing by 7, 0 through 6. Thus arithmetic modulus 7 has 7 numbers.

The results of addition modulus 7 cycle like addition of hours on a clock. When you get to the maximum value, start over again at 0.

$$\begin{aligned} 2 + 3 &= 5 \pmod{7} \\ 2 + 4 &= 6 \pmod{7} \\ 2 + 5 &= 0 \pmod{7} \\ 2 + 6 &= 1 \pmod{7} \\ &\vdots \end{aligned}$$

The complete addition table for addition mod 7:

Addition Mod 7 (3.1)

	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

What we have looked at examples of thus far is **modulus 7**, but there are moduli for every integer, with congruence defined exactly analogously to our definition for modulus 7

Addition in modular arithmetic is completely defined by

$$x \pmod{n} + y \pmod{n} = (x + y) \pmod{n} \quad (3.2)$$

Take a moment to ponder what equation 3.2 says: It says it doesn't matter where you perform the addition. You can do ordinary addition and then take the

modulus (the right-hand-side) or you can take the moduli of the two operands first and then do the addition modulus 7 following the table in (3.1). You get the same answer. An example. Let's do $39 + 75$ using ordinary addition first:

$$\begin{aligned} 39 + 75 &= 114 \\ 114 &= 16 \cdot 7 + 2 \\ (39 + 75) &\equiv 2 \pmod{7} \end{aligned}$$

Now the other way, modular conversion first:

$$\begin{aligned} 39 &= 5 \cdot 7 + 4 \\ 39 &\equiv 4 \pmod{7} \\ 75 &= 7 \cdot 10 + 5 \\ 75 &\equiv 5 \pmod{7} \\ 5 + 4 &\equiv 2 \pmod{7} \end{aligned}$$

Both computations yield 2 as the answer.

We write

$$g \mid n$$

when integer g divides integer n evenly. This is the same as saying:

$$g \equiv 0 \pmod{n}$$

The set of numbers that is congruent to $0 \pmod{7}$ can also be arranged in a series:

$$0, 7, 14, 21, 28, \dots$$

Notice that if you subtract any member of the series from any other you get a multiple of 7. Notice the same is true of the modulus 4 and modulus 3 series'. The following theorem expresses what we have just discovered. The proof shows why this is not an accident.

Theorem 3.1.1. $a \equiv b \pmod{n}$ if and only if $n \mid (a - b)$.

We first show that if $a \equiv b \pmod{n}$, then $n \mid (a - b)$. If $a \equiv b \pmod{n}$, then there is a remainder r such that:

$$\begin{aligned} (1) \quad a &= q_1 \cdot n + r \\ (2) \quad b &= q_2 \cdot n + r \end{aligned}$$

Subtracting 2 from 1, we get:

$$\begin{aligned} (a - b) &= q_1 \cdot n - q_2 \cdot n \\ (a - b) &= (q_1 - q_2) \cdot n \end{aligned}$$

Therefore $(a - b)$ is divisible by n .

Now the other direction. If $n \mid (a - b)$, then $a \equiv b \pmod{n}$. Without loss of generality we may express a and b as:

$$\begin{aligned} (1) \quad a &= q_1 \cdot n + r_1 \\ (2) \quad b &= q_2 \cdot n + r_2 \end{aligned}$$

Subtracting 2 from 1, we get:

$$\begin{aligned} (a - b) &= (q_1 \cdot n - q_2 \cdot n) + r_1 - r_2 \\ (a - b) &= (q_1 - q_2) \cdot n + r_1 - r_2 \end{aligned}$$

If $n \mid (a - b)$, then n must divide $r_1 - r_2$. By the definition of a remainder both r_1 and r_2 must be positive and less than n . But the difference of two positive numbers less than n must be less than n and greater than $-n$, and the only way such a number can be divisible by n is if that number is 0. Therefore:

$$\begin{aligned} r_1 - r_2 &= 0 \\ r_1 &= r_2 \end{aligned}$$

And therefore

$$a \equiv b \pmod{n}$$

Q.E.D.

3.1.2 Exercises

Formal definition of an equivalence relation. Consider the following relation R on the set of integers:

$$R = \{(x, y) \mid 7 \mid (x - y)\}$$

- (a) Show that R is an equivalence relation.
- (b) How many sets does R partition the set of integers into?
- (c) Show that:

$$R(x, y) \text{ if and only if } x \equiv y \pmod{7}$$

3.2 Grouphood

For this section review the axioms for groups.

3.2.1 Grouphood of complete residue systems under addition mod n

We claim the set

$$\{0, 1, 2, 3, 4, 5, 6\}$$

is a group under the operation of addition mod 7.

First, addition mod n is closed:

Addition Mod 7 (3.3)

	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

Second there is an identity element. What?

Next does each element have an inverse? What? Find the additive inverses mod 7:

$$\begin{aligned} 3^{-1} \pmod{7} &=? \\ 5^{-1} \pmod{7} &=? \\ 0^{-1} \pmod{7} &=? \end{aligned}$$

Note that we have no need of negative numbers in arithmetic mod 7. Our inverses can all be expressed as positive numbers p , such $0 < p < n$. For example, the inverse of 4 is 3:

$$3 + 4 = 0 \pmod{7}$$

It turns out that -4 is still congruent to the inverse of 4, since:

$$\begin{aligned} -4 &= -1 \cdot 7 + 3 \\ -4 &\equiv 3 \pmod{7} \end{aligned}$$

This is all quite fortunate since this is the only way the law of modular addition as laid out in (3.2) could be validated for $(-4 + 4)$. Doing ordinary addition followed by modular conversion, we have:

$$\begin{aligned} -4 + 4 &= 0 \\ 0 &\equiv 0 \pmod{7} \end{aligned}$$

Now the other way, $(-4 \bmod 7) + (4 \bmod 7)$

$$\begin{aligned} -4 &\equiv 3 \pmod{7} \\ 4 &\equiv 4 \pmod{7} \\ 3 + 4 &\equiv 0 \pmod{7} \end{aligned}$$

Both computations yield something congruent to $0 \pmod{7}$ as the answer. Whew.

We conclude this section with a brief discussion of **complete residue systems mod n**:

Consider arithmetic $\pmod{7}$ again, which we have defined in terms of the set of integers R :

$$R = \{0, 1, 2, 3, 4, 5, 6\}$$

We call R a **residue set** because it was defined in terms of remainders when dividing by 7 and residue is a technical term for remainder. What makes R adequate for doing arithmetic $\pmod{7}$? One key property is that we can “convert” any integer into some member of this set for purposes of doing arithmetic $\pmod{7}$. That is for any integer n , there is some $x \in R$ such that:

$$n \equiv x \pmod{7}$$

Thus each member of the residue set R represents an equivalence class of integers and there are enough such classes to handle all the integers. As we saw in Exercise 1 of Section 3.1.2 the equivalence classes can be defined directly in term of the following relation

$$\{(a, b) \mid n \mid (a - b)\}$$

This sorts the integers into 7 equivalence classes, each representable by a distinct member of R .

What if, instead, we had used the following integers?

$$R' = \{35, 36, 37, 38, 39, 40, 41\}$$

There is an important sense in which it wouldn't have made any difference. Each member of the first set has exactly one member of R' congruent to it with no leftovers. So R' exemplifies exactly the same set of equivalence classes as R . All that has changed is the “name” or representative member of the equivalence class. Crucially

$$42 + 7 \equiv 7 \equiv 0 \pmod{7}$$

is true whether you use R or R' . What we are really saying is that if you add a member of the “42” class to a member of the “7” class you get a member of the “0” class. Thus, the truth or falsity of a congruence statement is independent of which residue set you are using .

How would you compute with R' ? Well there is no doubt R is a lot more convenient, but you can convert any integer n into a congruent member x of R' as follows:

$$\begin{aligned} n \div 7 &= y \text{ Remainder } r \\ x &= r + 35 \end{aligned}$$

Thus in principle we can use any residue class we want. All we need is some way of making sure that pick out one and only one member from each of the equivalence classes to use as the representative.

We formalize this intuition with definition of a **complete residue system mod n** : any set of numbers that has the right number elements that fall into the right number of equivalence classes mod n :

Definition 3.2.1. *Complete Residue Systems mod n*

A Complete residue system mod n is a set of integers R such that

- 1. Completeness: For any integer x there is $y \in R$ such that $x \equiv y \pmod{n}$.*
- 2. Minimality: If $x, y \in R$, and $x \equiv y \pmod{n}$ then $x = y$*

So the two criteria are completeness, enough equivalence classes to cover the integers, and minimality, no distinct elements of R are congruent. It follows directly from the way we constructed R in the previous section that R is a complete residue system.

The next theorem gets at the idea that certain operations on a complete residue set like R are guaranteed to give use another complete residue set because, after performing the operation, we still have a set with one member from each equivalence class. This means we can prove modular arithmetic properties on “easy” set like R above and then transfer them to other sets like R' above.

Theorem 3.2.1. *Complete Residue systems by addition*

If R is a complete residue system mod n , then

$$\{j + x \mid x \in R\}$$

is a complete residue system mod n . We write this set $j + R$.

Let $a, b \in j + R$ be equal to $x + j$ and $y + j$, $x, y \in R$. Then

$$x + j \equiv y + j \tag{3.4}$$

if and only if¹

$$x \equiv y \tag{3.5}$$

Therefore $j + R$ has exactly as many equivalence classes as R . And since R , by assumption, was already minimal and complete, we are done.

Example. The set R' can be derived from R by addition as follows:

$$R' = \{35 + x \mid x \in R\}$$

Therefore since R is a complete residue system, R' is.

3.2.2 Modular Arithmetic Groups with Multiplication

For applications to cryptography, addition modulo n is not very helpful. What we need to look at is multiplication modulo n .

We define multiplication as an operation in modular arithmetic, governed by a law analogous to (3.2), the law governing modular addition:

$$x \bmod n \cdot y \bmod n = (x \cdot y) \bmod n \tag{3.6}$$

It turns out that it's a little trickier to find sets of integers that yield groups under modular multiplication than it was for modular addition. We start with an easy case. The set of numbers $\{1, 2, 3, 4, 5, 6\}$ is a group under multiplication mod 7.

Multiplication Mod 7 (3.7)

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

The standard notation for this group is Z_7^* . The Z stands for integers, the asterisk and subscript 7 tell us we are interested in the operation of multiplication mod 7.

Clearly the identity element in (3.7) is 1. Each row has a 1, so each element has a right inverse; and each column has a 1, so each element has a left inverse.

¹This follows from the cancellability theorem, Theorem 2.1.1, of course.

Note that if 0 were included in the set of elements under consideration, we would not have a group. Why not?

Now consider the case of multiplication mod 6. In particular, consider the multiplication table for 3 mod 6:

Multiplication Table for 3 Mod 6 (3.8)

	1	2	3	4	5
3	3	0	3	0	3

No 1's appear. Nothing congruent to 1 appears. The odd multiples of 3 are congruent to 3; the even multiples to 0. Clearly, since 3 has no inverse under multiplication mod 6, no set that includes it can be a group under that operation. In particular, the set

$$S = \{1, 2, 3, 4, 5\}$$

is not a group under that operation.

A consequence of this that equations involving multiplication mod 6 do not necessarily have unique solutions within S. For example, consider

$$3x \equiv 3 \pmod{6}$$

There is no legitimate inverse to multiply by to find a unique solution. This is not just a computational inconvenience. In fact there is no unique solution to find. As the multiplication table in (3.8) shows, 1, 3, and 5 are all solutions.

It turns out that if we restrict our attention to integers i , $0 < i < 6$, and exclude the trivial case of $\{1\}$, the only set that gives us a group under multiplication mod 6 is:

$$\{1, 5\}$$

Not coincidentally, it turns out that 1 and 5 are the only integers in this range that have inverses mod 6. The operation table looks like this:

	1	5
1	1	5
5	5	1

Note that 5 serves as its own inverse, correctly, since:

$$5 \cdot 5 = 25 \equiv 1 \pmod{6}$$

The trick to constructing a group for each modulus is to find a set of numbers such that all have an inverse.

Clearly for arbitrary n not every set of integers greater than 0 and less than n yields a group. Which sets form a group? Which ones don't? As the example of multiplication mod 6 suggests, this question is closely related to the question of what numbers have modular inverses. Is there an interesting answer to this question? [There might not be. The answer might always be: Do your times tables.] As it turns out there is an interesting answer.

A number y has an inverse modulo n if and only if y is relatively prime to n .

From this and some simple arithmetic, it will follow that for any modulus n , the set of numbers *relatively prime* to n and less than n form a multiplicative group we call

$$\mathbf{Z}_n^*.$$

As a special case, \mathbf{Z}_n^* for prime numbers includes all the integers from 1 to $(n - 1)$, which is why 1 through 6 made a group under multiplication mod 7. It is the grouphood of this kind of system, and the implied existence of inverses that comes with it, that will make such systems of interest for cryptography.

To show this will require some background laid out in the next few sections.

3.2.3 Exercises

1. Is:

$$(\{x \mid 0 \leq x < n\}, + \bmod n)$$

a sub-group of $(\mathbf{Z}, +)$ (the group of all integers under addition)? Why or why not?

2. Show that the set of integers relatively prime to 9 form a group under multiplication modulo 9.

$$\{1, 2, 4, 5, 7, 8\}$$

Don't forget to verify completeness.

3.3 Euclid's Algorithm and Euclid's Extended Algorithm

When

$$y \mid a \text{ and } y \mid b$$

3.3. EUCLID'S ALGORITHM AND EUCLID'S EXTENDED ALGORITHM⁴³

We say y is a common divisor of a and b . We call the greatest such number the *greatest common divisor* of a and b . We write:

$$\text{GCD}(y, n) = g$$

Examples:

1. 4 is the greatest common divisor of 52 and 96.

$$\begin{aligned} 52 &= 4 \cdot 13 = 2^2 \cdot 13 \\ 96 &= 32 \cdot 3 = 2^5 \cdot 3 \end{aligned}$$

2. 2 is a common divisor of 52 and 96, but not the greatest common divisor.
3. 1 is the greatest common divisor of 4 and 7.

When

$$\text{GCD}(y, n) = 1$$

we say y and n are relatively prime. This is also written:

$$y \perp n$$

In the next section we look at an important algorithm, Euclid's Algorithm, which finds the GCD of two numbers.

3.3.1 Euclid's Algorithm

There is an algorithm for discovering the GCD of two integers first written up by Euclid but apparently known for some time before that. This puppy has withstood the test of time. It is implemented today basically as he drew it up. As a special case it will reveal when two integers are relatively prime, because the GCD will then be 1.

In this and the following section we do the following:

1. Show the basic Euclid Algorithm which finds the GCD of two numbers y and n .
2. Show a more complicated extended version of the algorithm, which finds the GCD *and a way of expressing the GCD as a linear combination of y and n* . It turns out that this shows us how to define GCDs as the smallest positive integer that can be expressed as a linear combination of y and n .

3. This very directly provides a way of finding inverses in modular arithmetic,
4. Finally we prove y has an inverse $\pmod n$ if and only if $\text{GCD}(y, n) = 1$, which, as a corollary, will allow us to show $Z_{n,*}$ is a group.

Euclid's Algorithm. *Input:* a, b : Positive Integers
Output: g : an Integer

where g is the Greatest Common Divisor of a and b .

The algorithm will be recursive, which means it works by performing the same operation on simpler and simpler cases. Let's call the arguments of the procedure l and s (mnemonic for large and small, for reasons that will become clear). For purposes of this discussion we need to keep track of the progress of l , s and something called r (for remainder) across rounds, so we will call l , s and r in round 0 l_0 , s_0 and r_0 and l , s and r in round 1 l_1 , s_1 and r_1 , and so on. We start with l_0 set to a and s_0 set to b .

STEP ONE, ROUND n : First, check to see if $s_n = 0$. If it does, the answer $g = l_n$.

STEP TWO, ROUND n : If not, divide l_n by s_n . Note the remainder r_n . More precisely, find $r_n < s_n$ such that for some quotient q :²

$$\begin{aligned} (a) \quad q_n &= \lfloor l_n/s_n \rfloor \\ (b) \quad l_n &= q_n \cdot s_n + r_n \end{aligned} \tag{3.9}$$

Now get ready for the next round: Set l_{n+1} to be s_n and set s_{n+1} to r_n . Go back to step one for the next round..

That's it. That's all there is to it.

Figure 3.1 shows the computations for the case of 52 and 96. The algorithm yields the answer 4, which is correct.

Why does this work?

First convince yourself that the algorithm always terminates.³

² Equation (3.9) introduces a new bit of notation, $\lfloor r_1/r_2 \rfloor$. In general, for any real number x , $\lfloor x \rfloor$ is just x rounded down to the next lower integer; so, $\lfloor a/b \rfloor$ is just a divided by b with the remainder thrown away. For example, $\lfloor 7/4 \rfloor$ is 1, $\lfloor 9/4 \rfloor$ is 2, and so on.

³ As a reminder of how the terms quotient, dividend, and divisor work, recall that:

$$\text{quotient} = \frac{\text{dividend}}{\text{divisor}}$$

3.3. EUCLID'S ALGORITHM AND EUCLID'S EXTENDED ALGORITHM 45

Round	Parameters	Calculation
0	$l_0 = 96, s_0 = 52, r_0 = 44$	$96 = 1 \cdot 52 + 44$
1	$l_1 = 52, s_1 = 44, r_1 = 8$	$52 = 1 \cdot 44 + 8$
2	$l_2 = 44, s_2 = 8, r_2 = 4$	$44 = 5 \cdot 8 + 4$
3	$l_3 = 8, s_3 = 4, r_3 = 0$	$8 = 2 \cdot 4 + 0$
4	$l_4 = 4, s_4 = 0$	$g = 4$

Figure 3.1: Euclid's algorithm for the case of 52 and 96

At the end of a round n we prepare for the next round as follows::

$$\begin{array}{ll}
 l_{n+1} = s_n & l_{n+1} < l_n \\
 s_{n+1} = r_n & s_{n+1} < s_n \\
 r_{n+1} = l_{n+1} - q_{n+1} \cdot s_{n+1} & r_{n+1} < r_n = s_{n+1}
 \end{array}$$

The last two inequalities are true in virtue of the definition of remainders: a remainder is always smaller than the divisor that leaves it. The point here is that in each succeeding round the values of l , s and r are smaller than they were the round before. Since all three values are guaranteed to be integers s will sooner or later be 0 or 1. When s is 0 we're done. When it's 1 we're done in the next round, because 1 divides any number with remainder 0.

But why does it work? Consider $a = l_0$ and $b = s_0$. We are looking for an integer that divides both evenly. Without loss of generality we may express l_0 as:

$$l_0 = q_0 \cdot s_0 + r_0$$

Now any integer that divides s_0 must divide the first term on the right hand side evenly and therefore if it is to divide l_0 as well, it must divide the second term evenly too. Therefore we may reformulate our problem of finding the greatest integer that divides both l_0 and s_0 as a search for the greatest number that divides the smaller s_0 and r_0 . Now we recurse, writing our reformulation as an equation of the same form:

$$s_0 = q_1 \cdot r_0 + r_1$$

Or using the round 1 names for s_0 and r_0 :

$$l_1 = q_1 \cdot s_1 + r_1$$

Finally we are guaranteed to reduce the problem to one of finding the greatest integer that divides l_n and s_n where

$$l_n = q_n \cdot s_n + 0$$

But clearly that number is s_n . Q.E.D.

The argument is clear and ultimately very simple. What is really interesting is that after all these centuries no one has found a better shortcut for finding GCDs. You have to go through a chain of divisions, and pretty much Euclid's chain, to get the right answer in general.

A note on notation: Steps ONE and TWO of Euclid's algorithm may be reformulated as follows:

1. First, check to see if $l_n \equiv 0 \pmod{s_n}$. If so, $g = s$.
2. If not, set l_{n+1} to be s_n and set s_{n+1} to be $l_n \bmod s_n$ and go back to step one.

You will often see the algorithm presented this way. This is possible because it is the remainder and not the quotient that matters in each round and the quotient is precisely what the modular notation throws away. Thus in place of the calculation in round 2 of Figure 3.1 that

$$44 = 5 \cdot 8 + 4$$

in which 5 is the quotient, we would write simply:

$$44 \equiv 4 \pmod{8}$$

In the next section, however, we present an extended version of Euclid's algorithm in which the values of the quotients are not thrown away. To make the relationship between the extended and unextended algorithm clearer, we present the calculations in Figure 3.1 using quotients.

3.3.2 Exercises

1. Find the GCD of 108 and 42.

3.3.3 Euclid's Extended Algorithm

In this section we prove the following theorem.

The GCD of y and n is expressible as a linear combination of λ and μ .

$$g = \lambda y + \mu n$$

where λ and μ are integers.

3.3. EUCLID'S ALGORITHM AND EUCLID'S EXTENDED ALGORITHM 47

Note that this property is not supposed to be obvious.

$$\begin{aligned} \text{(a)} \quad & 1 = -3 \cdot 9 + 4 \cdot 7 \\ \text{(b)} \quad & 4 = -5 \cdot 9 + 7 \cdot 7 \end{aligned}$$

In (a) 1 is expressible as a linear combination of 9 and 7. In fact 1 is the GCD of 9 and 7. In (b) 4 is expressible as a linear combination of 9 and 7, but 4 is not the GCD of 9 and 7; in fact, it isn't even a divisor of 9 and 7. So the connection between being expressible as a linear combination and being a GCD is obscure.

In our introductory discussion of Euclid's algorithm we noted there were three numbers involved in the computation performed each round and we named them l , s and r . One problem with this is that the same number gets three different names as we progress through the rounds. The number that plays the remainder role in round 0 is r_0 , but that same number plays the s role in round 1 (s_1) and then the l role in round 2 (l_2). (see Figure 3.1). For purposes of proving something about Euclid's algorithm it will be useful to consistently use one name for each value. Thus, in the following discussion we look at things only from the l point of view. The two arguments a and b will get set to be l_0 and l_1 and we will refer to the first remainder only as l_2 . Consistent with this policy, we will say the algorithm terminates when $l_i = 0$ and the GCD we are computing is l_{i-1} (Compare Figure 3.1).

Theorem 3.3.1. *If g is the Greatest Common Divisor of a and b then there exist integers λ and μ such that⁴*

$$g = \lambda \cdot a + \mu \cdot b \tag{3.10}$$

Sketch of proof. We are going to prove for each of the l_i in Euclid's algorithm that it can be expressed in the form of Equation 3.10. Then, in particular, that will be true for the last l_i but one, which is the GCD of a and b . We show it first for the first two l_i s, l_0 and l_1 . These are what we call the base cases below. Then we show that if it holds for l_{i-1} and l_i , it must hold for l_{i+1} .

STEP ONE: The base cases: l_0 and l_1 . These can be expressed in the form of Equation 3.10 as follows:

$$l_0 = a; \quad l_1 = b$$

	λ	μ
$l_0 = 1 \cdot a + 0 \cdot b$	1	0
$l_1 = 0 \cdot a + 1 \cdot b$	0	1

STEP TWO: We now need a general way to find values for λ_{i+1} and μ_{i+1} given values for l_{i-1} , l_i and λ_i and μ_i . First we restate the procedure of Euclid's

⁴ The fancy terminology for this is that g is a linear combination of λ and μ .

algorithm for finding l_{i+1} in terms of the previous two l s:

$$\begin{aligned} (a) \quad q_i &= \lfloor l_{i-1}/l_i \rfloor \\ (b) \quad l_{i+1} &= l_{i-1} - q_i \cdot l_i \end{aligned} \tag{3.11}$$

Then λ_{i+1} and μ_{i+1} are computed as follows:

$$\begin{aligned} (a) \quad \lambda_{i+1} &= \lambda_{i-1} - q_i \cdot \lambda_i \\ (b) \quad \mu_{i+1} &= \mu_{i-1} - q_i \cdot \mu_i \end{aligned} \tag{3.12}$$

Again, this is not supposed to be obvious, but it works. Here's the proof. First we are assuming that the previous two l 's can already be expressed in the form of Equation 3.10. That is:

$$\begin{aligned} (a) \quad l_{i-1} &= \lambda_{i-1} \cdot a + \mu_{i-1} \cdot b \\ (b) \quad l_i &= \lambda_i \cdot a + \mu_i \cdot b \end{aligned} \tag{3.13}$$

What we now need to show is that, given (3.11), (3.12), and (3.13),

$$l_{i+1} = \lambda_{i+1} \cdot a + \mu_{i+1} \cdot b \tag{3.14}$$

This takes a little algebraic manipulation:

$$\begin{aligned} l_{i+1} &= l_{i-1} - q_i \cdot l_i && \text{Equation 3.11b} \\ &= \lambda_{i-1} \cdot a + \mu_{i-1} \cdot b - q_i(\lambda_i \cdot a + \mu_i \cdot b) && \text{Equation 3.13a,b} \\ &= a(\lambda_{i-1} - q_i \lambda_i) + b(\mu_{i-1} - q_i \cdot \mu_i) \\ &= \lambda_{i+1} \cdot a + \mu_{i+1} \cdot b && \text{Equation 3.12a,b} \end{aligned}$$

Which is what wanted to show.

Corollary 3.3.1. *Let g be the greatest common divisor of a and b . Then g is the smallest positive integer such that for some integers λ and μ :*

$$g = \lambda \cdot a + \mu \cdot b$$

Step One: Theorem 3.3.1 is a proof that g is a positive integer that can be expressed as a linear combination of a and b .

Step Two: Suppose there were a smaller positive integer l , $0 < l < g$, such that

$$l = \lambda_1 \cdot a + \mu_1 \cdot b$$

By assumption $g \mid a$ and $g \mid b$. Therefore g divides l :

$$\frac{l}{g} = \frac{\lambda_1 \cdot a}{g} + \frac{\mu_1 \cdot b}{g}$$

$\frac{l}{g}$ is the sum of two integers and therefore must be an integer. But then, l is a positive integer such that g divides it. But then $l < g$ is impossible. Therefore there cannot exist such an l and therefore g must be the least positive integer expressible as a linear combination of a and b . Q.E.D.

3.3. EUCLID'S ALGORITHM AND EUCLID'S EXTENDED ALGORITHM 49

l_1, l_2	$\left[\begin{array}{c ccc} i & l & \lambda & \mu \\ \hline 1 & 96 & 1 & 0 \\ 2 & 52 & 0 & 1 \end{array} \right]$	$l_1 = 96; \lambda_1 = 1; \mu_1 = 0$ $l_2 = 52; \lambda_2 = 0; \mu_2 = 1$ $96 = (1 \cdot 96) + (0 \cdot 52)$ $52 = (0 \cdot 96) + (1 \cdot 52)$
l_3	$\left[\begin{array}{c ccc} i & l & \lambda & \mu \\ \hline 1 & 96 & 1 & 0 \\ 2 & 52 & 0 & 1 \\ \hline 3 & 44 & 1 & -11 \end{array} \right]$	$q_2 = \lfloor l_1/l_2 \rfloor = \lfloor 96/52 \rfloor = 1$ $l_3 = l_1 - q_2 \cdot l_2 = 96 - (1 \cdot 52) = 44$ $\lambda_3 = \lambda_1 - q_2 \cdot \lambda_2 = 1 - (1 \cdot 0) = 1$ $\mu_3 = \mu_1 - q_2 \cdot \mu_2 = 0 - (1 \cdot 1) = -1$ $44 = (1 \cdot 96) + (-1 \cdot 52)$
l_4	$\left[\begin{array}{c ccc} i & l & \lambda & \mu \\ \hline 2 & 52 & 0 & 1 \\ 3 & 44 & 1 & -1 \\ \hline 4 & 8 & -1 & 2 \end{array} \right]$	$q_3 = \lfloor l_2/l_3 \rfloor = \lfloor 52/44 \rfloor = 1$ $l_4 = l_2 - q_3 \cdot l_3 = 52 - (1 \cdot 44) = 8$ $\lambda_4 = \lambda_2 - q_3 \cdot \lambda_3 = 0 - (1 \cdot 1) = -1$ $\mu_4 = \mu_2 - q_3 \cdot \mu_3 = 1 - (1 \cdot -1) = 2$ $8 = (-1 \cdot 96) + (2 \cdot 52)$
l_5	$\left[\begin{array}{c ccc} i & l & \lambda & \mu \\ \hline 3 & 44 & 1 & -1 \\ 4 & 8 & -1 & 2 \\ \hline 5 & 4 & 6 & -11 \end{array} \right]$	$q_4 = \lfloor l_3/l_4 \rfloor = \lfloor 44/8 \rfloor = 5$ $l_5 = l_3 - q_4 \cdot l_4 = 44 - (5 \cdot 8) = 4$ $\lambda_5 = \lambda_3 - q_4 \cdot \lambda_4 = 1 - (5 \cdot -1) = 6$ $\mu_5 = \mu_3 - q_4 \cdot \mu_4 = (-1) - (5 \cdot 2) = -11$ $4 = (6 \cdot 96) + (-11 \cdot 52)$ Answer
l_6	$\left[\begin{array}{c ccc} i & l & \lambda & \mu \\ \hline 4 & 8 & -1 & 2 \\ 5 & 4 & 6 & -11 \\ \hline 6 & 0 & -13 & 24 \end{array} \right]$	$q_5 = \lfloor l_4/l_5 \rfloor = \lfloor 8/4 \rfloor = 2$ $l_6 = l_4 - q_5 \cdot l_5 = 8 - (2 \cdot 4) = 0$ $\lambda_6 = \lambda_4 - q_5 \cdot \lambda_5 = (-1) - (2 \cdot 6) = -13$ $\mu_6 = \mu_4 - q_5 \cdot \mu_5 = 2 - (2 \cdot -11) = 24$ $0 = (-13 \cdot 96) + (24 \cdot 52)$ Terminate

Figure 3.2: Example of Euclid's Extended Algorithm with $a = 96$ and $b = 52$

Figure 3.2 shows the calculations for Euclid's algorithm using the example of 96 and 52, whose GCD is 4. The step marked l_1, l_2 is the initialization step in which both l_1 and l_2 are set, along with their corresponding initial λ s and μ s.

Each succeeding section of the figure is marked with the l -value computed in that step. In general each step shows the process of computing l_{i+1} given l_i and l_{i-1} . In each case the first computation involves computing the quotient q_i from the rounded-down ratio of l_{i-1} to l_i . Then the next l_{i+1} is the remainder of l_{i-1} divided by l_i (or $l_{i-1} \bmod l_i$). This is computed by subtracting $l_i \cdot q_i$ from l_{i-1} . Similarly μ_{i+1} is computed by subtracting $\mu_i \cdot q_i$ from μ_{i-1} and λ_{i+1} by subtracting $\lambda_i \cdot q_i$ from λ_{i-1} .

The answer is the l_i that divides l_{i-1} exactly. This is found in the l_5 section when l_5 is determined to be 4. The algorithm terminates with l_6 in the next step when l_6 is determined to be 0. Thus the answer is the remainder that causes the next remainder to be 0.

The computations determining each l , λ and μ are shown in the right hand column of Figure 3.2. The calculation below the line in figure 3.2 shows that each l_i can be expressed as a linear combination of 96 and 52, using λ_i and μ_i as the linear weights.

In particular the answer, $l_5 (= 4)$, is expressed as a linear combination, which is what Theorem 3.3.1 shows must be possible. Interestingly, since $l_6 (= 0)$ bears the same relationship to previous l_i s as all the others, it too can be expressed as a linear combination of its λ and μ , though this fact is not very useful.

3.3.4 Exercises

1. Find the GCD of 108 and 42 and a linear combination that expresses it, using Euclid's extended algorithm.
2. Find the inverse of 5 mod 16 using Euclid's extended algorithm. Note 1: You can do this by clever guessing or exhaustive search, but use the algorithm and show your work. Note 2: Your answer should be a positive number i such that $0 < i < 16$. Note 3: Check your answer. That is, verify that $5 \cdot 5^{-1} \equiv 1 \pmod{16}$.
3. Find the inverse of 53 mod 175 using Euclid's extended algorithm. Pretty much the same notes as in the first problem, except that this time your answer should be a positive number i such that $0 < i < 175$.

3.4 Theorems about modular inverses

Theorem 3.4.1. $y \perp n$ if and only if y has an inverse mod n .

Part I: If $y \perp n$, then y has an inverse mod n . Assume $y \perp n$. By Theorem 3.3.1, we know that there exist integers λ, μ , such that

$$\lambda y + \mu n = 1.$$

But then:

$$\begin{aligned} \mu n &= 1 - \lambda y \\ n &| 1 - \lambda y \\ \lambda y &\equiv 1 \pmod{n} \quad \text{Theorem 3.1.1} \end{aligned}$$

Therefore y has an inverse mod n , namely λ .

Part II: If y has an inverse mod n , then $y \perp n$.

Assume y^{-1} exists. Then for some λ ,

$$\lambda y \equiv 1 \pmod{n}$$

That is for some μ

$$\begin{aligned} \mu n &= \lambda y - 1 \\ \mu n - \lambda y &= -1 \\ -\mu n + \lambda y &= 1 \end{aligned}$$

According to Corollary 3.3.1, the GCD of y and n is the smallest positive number that can be expressed as a linear combination of y and n . But there are no positive integers smaller than 1. Therefore 1 must be the GCD of y and n . Q.E.D.

Chapter 4

Group Properties of Multiplicative Groups

4.1 Grouphood of Z_n^*

Recall that Z_n^* is pairs the set of positive integers relatively prime to n and less than n with the operation of multiplication mod n . In this section we prove the following theorem.

Theorem 4.1.1. *Grouphood of Z_n^**

Z_n^ is a group.*

Proof: *The operation of Z_n^* is multiplication mod n , which is associative, and the identity element of Z_n^* is 1. By Theorem 3.4.1, every element of Z_n^* has an inverse under multiplication mod n , and also by Theorem 3.4.1, each of those inverses must be relatively prime to n and therefore in Z_n^* .*

What remains to be shown is closure: The modulus of the product of two numbers relatively prime to n is relatively prime to n . This is left as an exercise.

4.1.1 Exercises

1. Complete the proof of Theorem 4.1.1 by showing that the modulus of the product of two numbers relatively prime to n is relatively prime to n . In

symbols:

$$\begin{aligned} \text{If:} & \quad i \perp n \\ & \quad j \perp n \\ \text{Then:} & \quad (i \cdot j) \bmod n \perp n \end{aligned}$$

Hint: First show that if $p \equiv q \pmod n$ then $p \perp n$ if and only if $q \perp n$. It may help to represent p and q as:

$$\begin{aligned} p &= k_1 \cdot n + r \quad 0 \leq r < n \\ q &= k_2 \cdot n + r \end{aligned} \tag{4.1}$$

Then show $i \cdot j \perp n$ if $i, j \perp n$.

4.2 Subgroups and Cosets of Multiplicative Groups

The Z_n^* groups are the groups that are important for cryptographic applications. In this section we illustrate some important structural properties of these groups, focusing on cyclic subgroups. Recall that these groups are formed by iterated applications of the group operation to a single element, *raising to a power*.

As an example let us return to Z_7^* . It turns out that Z_7^* is a cyclic group for which 3 is a generator, since every element of the group can be expressed as a power of 3 (Cyclic groups are introduced in definition 2.1.4):

$$\begin{aligned} 1 &= 3^6 \pmod 7 \\ 2 &= 3^2 \pmod 7 \\ 3 &= 3^1 \pmod 7 \\ 4 &= 3^4 \pmod 7 \\ 5 &= 3^5 \pmod 7 \\ 6 &= 3^3 \pmod 7 \end{aligned}$$

Note that not every member of the group generates it. For example, 2 is not a generator for the entire group. It generates only the set $\{1, 2, 4\}$

$$\begin{aligned} 2 &= 2^1 \pmod 7 \\ 4 &= 2^2 \pmod 7 \\ 1 &= 2^3 \pmod 7 \\ 2 &= 2^4 \pmod 7 \\ 4 &= 2^5 \pmod 7 \\ 1 &= 2^6 \pmod 7 \end{aligned}$$

Thus, according to our definition of *the order of an element* in Section 2.3.2, 7 has order 6 and 2 has order 3. Note that Z_7^* has 6 elements and both 2 and 6 divide 6 evenly. Is this an accident?

More precisely, Is it the case that for any element a of any Z_n^* , $\text{ord}(a)$, the order of a , divides $|Z_n^*|$?

Recall that for any finite group \mathbf{G} , and any $a \in \mathbf{G}$, the set:

$$\mathbf{H} = \{a^i \mid \text{for some integer } i\},$$

is a subgroup of \mathbf{G} . The question of possible orders of elements turns out to be a special case of the question of possible orders of subgroups. Is the order of a group divisible by the order of its subgroups?

In the next section we will prove a theorem (LaGrange's Theorem) which answers this question in the affirmative.

The proof will hinge on the notion of a coset, which is a set that can be constructed once a subgroup is given:

Definition 4.2.1. *Coset*

Let (\mathbf{G}, \circ) be a group and $\mathbf{H} \subseteq \mathbf{G}$. For $a \in \mathbf{G}$,

$$a \circ \mathbf{H} \stackrel{\text{def}}{=} \{a \circ h \mid h \in \mathbf{H}\}$$

is called a (left) coset of \mathbf{H} . We say a yields the coset $a \circ \mathbf{H}$.

Example Let \mathbf{H} be the set generated by 2 in Z_7^* :

$$\mathbf{H} = \{1, 2, 4\}$$

Then we may use 3 to generate the following coset:

$$3 \cdot \mathbf{H} = \{3, 5, 6\}$$

Note that this coset is both the same size as the subgroup \mathbf{H} and disjoint from it. These properties too are not accidents. We will prove LaGrange's theorem by showing that any subgroup gives rise to a family of cosets, all disjoint from it and each other, and all of the same size as the subgroup.

We begin by showing that cosets have the same cardinality as the group that generates them..

Lemma 4.2.1. *Coset Cardinality*

Since the definition of coset requires every member of $a \circ \mathbf{H}$ to be the unique result of combining a with some member of \mathbf{H} , we have:

$$|a \circ \mathbf{H}| \leq |\mathbf{H}|$$

The only way that

$$|a \circ \mathbf{H}| < |\mathbf{H}|$$

can be true is if two distinct elements of \mathbf{H} call them b, c , give the same result when combined with a :

$$a \circ b = a \circ c \quad \text{Convergence}$$

But the cancellation theorem (Theorem 2.1.1) tells us that in this case, b and c cannot be distinct. Therefore, the cardinality of the coset $a \circ \mathbf{H}$ must be the same as the cardinality of \mathbf{H} .

4.3 LaGrange's Theorem

We now show:

Lemma 4.3.1. *Coset Partitioning* Let $a, b \in \mathbf{G}$ and let \mathbf{H} be a subgroup of \mathbf{G} . Then either:

$$a \circ \mathbf{H} = b \circ \mathbf{H}$$

or

$$(a \circ \mathbf{H}) \cap (b \circ \mathbf{H}) = \emptyset$$

Proof Suppose that

$$(a \circ \mathbf{H}) \cap (b \circ \mathbf{H}) \neq \emptyset \tag{4.2}$$

Then there exists $s, t \in \mathbf{H}$ such that

$$a \circ s = b \circ t \tag{4.3}$$

But then:

$$a = b \circ t \circ s^{-1} \tag{4.4}$$

And for an arbitrary x

$$a \circ x = b \circ t \circ s^{-1} \circ x \tag{4.5}$$

Now by the inverse axiom and closure, for any $x \in \mathbf{H}$:

$$t \circ s^{-1} \circ x \in \mathbf{H}$$

And therefore, for any $x \in \mathbf{H}$:

$$b \circ t \circ s^{-1} \circ x \in b \circ \mathbf{H} \tag{4.6}$$

And from (4.5) and (4.6) it follows that for any $x \in \mathbf{H}$.

$$a \circ x \in b \circ \mathbf{H} \quad (4.7)$$

Thus

$$a \circ \mathbf{H} \subseteq b \circ \mathbf{H} \quad (4.8)$$

From (4.3) we also have

$$b = a \circ s \circ t^{-1} \quad (4.9)$$

And from steps parallel to (4.5) to (4.7) we have for any $y \in \mathbf{H}$:

$$b \circ y = a \circ s \circ t^{-1} \circ y \in b \circ \mathbf{H} \quad (4.10)$$

And therefore:

$$b \circ \mathbf{H} \subseteq a \circ \mathbf{H} \quad (4.11)$$

And from (4.8) and (4.11) we have:

$$a \circ \mathbf{H} = b \circ \mathbf{H} \quad (4.12)$$

So we have shown that if two cosets of \mathbf{H} are not disjoint (4.2), then they are equal (4.12).

LaGrange's Theorem. Let \mathbf{H} be a subgroup of finite group \mathbf{G} . Then the cardinality of \mathbf{H} evenly divides the cardinality of \mathbf{G} .

Proof Let \mathbf{G} have cardinality n . Now each member a_i of \mathbf{G} will generate a coset of \mathbf{H} :

$$a_i \circ \mathbf{H}$$

And moreover,

$$a_i \in a_i \circ \mathbf{H}$$

That follows because the identity element $e \in \mathbf{H}$ and

$$a \circ e = a$$

Therefore if m is the number of distinct cosets and we pick one representative member a_i from each of the m cosets, we have:

$$\mathbf{G} = a_1 \circ \mathbf{H} \cup a_2 \circ \mathbf{H} \cup \cdots \cup a_m \circ \mathbf{H}$$

Now by Lemma 4.3.1 each element can occur in at most one of the m cosets. By Lemma 4.2.1, each of those cosets has cardinality $|\mathbf{H}|$. Therefore:

$$m \cdot |\mathbf{H}| = n = |\mathbf{G}|$$

Which is what we wanted to show.

Corollary 4.3.1. *Consider the definition of the order of an element of a group \mathbf{G} given in Exercise 2 of Section 2.3. Every element with a finite order n is the generator of a subgroup of order n . Therefore, by Lagrange's Theorem, the order of any element of \mathbf{G} divides the order of \mathbf{G} .*

Example: Returning to the case of \mathbf{Z}_7^* and the subgroup generated by 2, we see that $\text{ord}(2) = 3$ (the order of 2) divides $|\mathbf{Z}_7^*|$ and $\text{ord}(6) = 6$ divides $|\mathbf{Z}_7^*|$, as required by Lagrange's theorem.

4.4 Fermat's Little Theorem and Euler's Theorem

4.4.1 Euler's Totient Function ϕ

Euler's Totient function ϕ is defined as follows:

$\phi(n)$ is the number of integers y , $1 < y \leq n$ such that $y \perp n$

There is one special case for which $\phi(n)$ is trivial. This is when n is prime. All numbers are relatively prime to a prime, so the the number of numbers less than n and relatively prime to n is $(n - 1)$.

The function $\phi(n)$ is significant because, for any n , $\phi(n)$ is the order (the number of elements) of \mathbf{Z}_n^* , the multiplicative group of integers that are relatively prime to n , discussed in Section 3.2.2. Such groups are central to the proof of Euler's Theorem.

For clues on how to compute $\phi(n)$ for composite numbers, see Appendix 6.1.

4.4.2 Fermat's Little Theorem

Here is what is called Fermat's Little Theorem: If p is prime,

$$a^{p-1} \equiv 1 \pmod{p}$$

This will fall out as a special case of Euler's Theorem. We prove that in the next section.

4.4.3 Euler's Theorem

Euler's Theorem will lead directly to a cryptographic application.

If a is relatively prime to n ,

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

where ϕ is Euler's totient function.

The first observation to make is that Fermat's Little Theorem is a special case of this. In the special case of a prime number, p , all the integers less than p are relatively prime to it, so the value of $\phi(p)$ is $p - 1$.

The next observation is that we're in a position to understand the restriction to an a that is relatively prime to n . If a raised to some power $\phi(n)$ in a multiplicative group is 1, then a has a multiplicative inverse mod n , namely $a^{\phi(n)-1}$:

$$a \cdot a^{\phi(n)-1} = a^{\phi(n)} = 1$$

But only numbers relatively prime to n have inverses. We have already proved (Exercise 2 of Section 2.3.1) that, in a finite group, the inverse of a is always a power of a . For the special case of multiplicative groups mod n , Euler's theorem gives us a specific power, $\phi(n) - 1$, that is guaranteed to be an inverse.

Proof: We prove this by considering Z_n^* , the multiplicative group introduced in Section 3.2.2, whose grouphood was proved in Theorem 4.1.1.

Recall that this is the group consisting of all the integers i , $0 < i < n$, i relatively prime to n . Therefore $|Z_n^*|$ is $\phi(n)$. Since a is relatively prime to n , there is some $i \in Z_n^*$ such that, $a \equiv i$. The Corollary of Lagrange's theorem, Corollary 4.3.1, tells us that the order of i , $\text{ord}(i)$, divides $\phi(n)$. So there is some integer k , such that

$$\text{ord}(i) \cdot k = \phi(n)$$

and

$$a^{\phi(n)} \equiv i^{\text{ord}(i) \cdot k} = i^{\text{ord}(i)^k} \equiv e^k \equiv 1 \pmod{n}$$

So this follows quite straightforwardly from Lagrange's Theorem.

Example: Returning to the case of Z_7^* , recall that $\text{ord}(2) = 3$. Therefore:

$$2^3 \cong 1 \pmod{7}$$

$\phi(7) = 6$ and

$$2^6 = (2^3)^2 = 1^2 = 1 \pmod{7}$$

As noted at the beginning of this section, Euler's theorem leads directly to a cryptographic application. The important application of Euler's Theorem is in the RSA protocol, which achieved fame as the first public key encryption scheme. We will discuss RSA in the next chapter.

In the meantime as a preparation, we present the following protocol, which we call **Euler's Code**. Euler's Code is somewhat simpler than RSA, but utilizes the same basic ideas. It lacks the critical element that made RSA famous because it is not a public key scheme. Although distinct keys are used for encryption and decryption, each is easily deduced from the other. Thus, practically speaking, this functions indistinguishably from an ordinary symmetric key code, in which a single key is used for both encryption and decryption.

1. Choose a large random prime number p
2. Randomly choose an encryption key e such that e and $(p-1)$ are relatively prime. Recall that for a prime, p , $(p-1)$ is the value of $\phi(p)$, the number of integers relatively prime to n .
3. Use the extended Euclidean Algorithm to compute the inverse of $e \bmod \phi(p)$, d . That is, find d , $1 < d < \phi(n)$, such that:

$$e \cdot d \equiv 1 \pmod{\phi(n)}$$

d is the decryption key.

4. The protocol will allow us to encrypt a message smaller than p . Longer messages will have to be broken up into blocks with each block encoded separately. Here is the encoding of a message m :

$$c = m^e \bmod n$$

5. The decoding of the message is as follows:

$$m = c^d \bmod n$$

That's all there is to it.

Why does it work? Clearly, it had better be the case that $c^d = m$. And it is. First let's just review how the pieces were put together:

$$c^d = (m^e)^d = m^{e \cdot d} \bmod n \tag{4.13}$$

Because e and d are inverses $\bmod \phi(n)$, there is some k such that:

$$e \cdot d = k \cdot (p-1) + 1$$

Rewriting the last term in (4.13) we have:

$$m^{e \cdot d} = m^{k \cdot (p-1) + 1} = m \cdot m^{k \cdot (p-1)} = m \cdot (m^{(p-1)})^k \pmod n \quad (4.14)$$

Because p is prime m is relatively prime to p and therefore Euler's theorem applies in this case. So we have:

$$m^{(p-1)} = m^{\phi(n)} \equiv 1 \pmod n$$

so the last term in (4.14) reduces as follows:

$$m \cdot (m^{(p-1)})^k \equiv m \cdot 1^k \equiv m \cdot 1 \equiv m \pmod n \quad (4.15)$$

Which is what we wanted to show.

So encryption and decryption work the way they should. The security of the algorithm is based on the fact that it is in general quite difficult to take the e -th root of a number $\pmod n$ if e is sufficiently large. One reason why no one uses such an algorithm for encryption is that is far less efficient than most existing symmetric algorithms, and generally speaking, efficiency is a pressing concern for encryption software. Thus, without the extra benefits public keys give, it is not worthwhile encrypting this way.

Example: Let's try an example with small numbers. In fact, let's set p equal to $\pmod{11}$; $\phi(11) = 10$. For our public key, we choose an exponent relatively prime to 10, 3. So to find our private key we need to find

$$3^{-1} \pmod{10}$$

One very simple way to find the inverse is to use Euler's Theorem. We know:

$$3^{\phi(10)} \equiv 1 \pmod{10}$$

What is $\phi(10)$. This is the number of integers less than 10 and relatively prime to 10: We write Z_{10}^* for the set of integers relatively prime to 10

$$\begin{aligned} Z_{10}^* &= \{1, 3, 7, 9\} \\ |Z_{10}^*| &= 4 \end{aligned}$$

So $\phi(10) = 4$. So, by Euler's Theorem:

$$3^4 \equiv 1 \pmod{10}$$

Therefore

$$\begin{aligned} 3 * 3^3 &\equiv 1 \pmod{10} \\ 3^{-1} &= 3^3 \pmod{10} = 27 \equiv 7 \pmod{10} \end{aligned}$$

We could solve the same problem a different way using Euclid's Extended Algorithm. We have:

$$\begin{array}{r}
 l_0, l_1 \quad \left[\begin{array}{ccc} l & \lambda & \mu \\ 10 & 1 & 0 \\ 3 & 0 & 1 \end{array} \right] \quad \begin{array}{l} l_0 = 10, l_1 = 3 \\ \lambda_0 = 1, \mu_0 = 0; \lambda_1 = 0, \mu_1 = 1; \\ \hline 10 = (1 \cdot 10) + (0 \cdot 3) \\ 3 = (0 \cdot 10) + (1 \cdot 3) \end{array} \\
 \hline
 l_2 \quad \left[\begin{array}{ccc} l & \lambda & \mu \\ 10 & 1 & 0 \\ 3 & 0 & 1 \\ \hline 1 & -3 & 1 \end{array} \right] \quad \begin{array}{l} q_2 = \lfloor 10/3 \rfloor = 3 \\ l_2 = 10 - (3 \cdot 3) = 1 \\ \lambda_2 = 0 - 1 \cdot 3 = -3 \\ \mu_2 = 1 - 0 \cdot 3 = 1 \\ \hline 1 = 1 \cdot 10 + (-3 \cdot 3) \end{array}
 \end{array}$$

So we have

$$3^{-1} = -3 \equiv 7 \pmod{10}$$

Verifying:

$$3 \cdot 7 = 21 \equiv 1 \pmod{10}$$

So the inverse of 3 is 7. Same answer as before. Thank goodness.

So we have encryption key 3 and decryption key 7. So let us encrypt a message one letter long choosing d (the 4th letter, which we represent with the number 4; note that we picked a letter whose numerical representation was less than 11). Encoding then is:

$$4^3 \pmod{11} = 64 \equiv 9 \pmod{11}$$

Now let's decrypt.

$$\begin{aligned}
 9^7 &= 9^3 \cdot (9)^4 \pmod{11} \\
 &\equiv 3 \cdot 5 \\
 &\equiv 4
 \end{aligned}$$

Which, exactly as desired, is the representation for d .

4.4.4 Exercises

- Using 17 as the value for p encrypt and decrypt the letter j using Euler's Code. Use either 3 or 4 for your encryption key, whichever is better. Defend your choice. Show the calculations determining your decryption key and verify that it is an inverse of your encryption key in the right modulus.

Chapter 5

Cryptography and Public Keys

5.1 Subject Matter of Cryptography

The general subject matter of cryptography is communication over a problematic channel. The channel may be insecure in the sense that any communication may be overheard, or insecure in the sense that communications may be arbitrarily altered before reaching their destination, and, in addition, the participants in the communication may not be trustworthy. They may repudiate important warrants at a later time. Assurances of various sorts would be valuable.

- **Confidentiality:** assurance that only the participants will know the contents of their communication. Business or government secrets.
- **Integrity:** assurance that the message sent is exactly the message received by the intended participant.
- **Authenticity:** assurance that the message really comes from who it says it comes from
- **Non-repudiation:** assurance that the real sender cannot deny a file was sent.
- **Content Commitment without Content Disclosure:** assurance that a certain content, say the answer to a particular question,, is retrievable with disclosing what that content is.

With regard to the last item, a few motivating remarks may be helpful. In certain circumstances to be discussed below, it is important for one participant in a communication to commit him or her self to a particular piece of content, say a bit that might be 1 or 0, without revealing what that bit is. This may be because of timing issues (the commitment must come early but the disclosure must come late) or it may be because the content is to be revealed only in certain extraordinary circumstances.

The general subject matter of modern cryptography then is verification and securing of communications over channels that are not secure and cannot be assumed to transmit information reliably and between participants who do not necessarily trust one another. Sounds like life.

Several important areas of computer security have been left out of the above list because they do not create a need for public keys. Thus for example a central concern for cryptographers is securing stored data or computer systems. Both these tasks can be done with single key systems and symmetric encryption algorithms. Thus for example authentication for almost all multi-user systems uses password programs that protect stored passwords by encrypting them.

However it turns out that the other important idea of this chapter, a one-way function, does play a role in ordinary system and data security. We return to this point below.

It also turns out that public key protocols do sometimes play a role in user authentication for system access. We return to this point below as well.

5.2 One-way functions

In this section we introduce one of the most important concepts of modern cryptography, the one-way function.

One way functions play a central role in public key cryptography but in this section we try to motivate the idea within the general context of cryptographic concerns, independently of keys.

5.2.1 Fair play: A coin-tossing game problem

Our first example is a simple case in which the problem is trying to secure content commitment without content disclosure. It is useful in setting the scene because it provides a scenario in which the cryptographic problem is not stereo-

typical. There is no issue of secrecy per se. Thus we have a scenario involving two characters Alice and Bob in a situation requiring information to be communicated, but there is no evil adversary (usually Eve) to be thwarted. Rather there is a larger concern of trust.

Alice and **Bob** (the main characters of our little drama) play a game of heads-or-tails.

Version 1: Alice flips coin. Bob calls it in the air. If Bob gets it right he wins. If not, Alice wins.

Issues:

- Fair coin
- Bob calls coin at right time
- Alice and Bob agree on outcome

Item 3 sounds silly.

But now add: Alice and Bob are playing over the phone.

How do we do this reliably, assuming Alice and Bob can't trust each other completely?

5.2.2 A solution: A one-way function

Wanted: a function f such that:

- For any x , $f(x)$ is pretty easy to compute
- For any $f(x)$, it is impossible to determine what x is.
- For any x and y , if $x \neq y$, then $f(x) \neq f(y)$

Never mind for now whether such a thing exists. Suppose we had one. Then agree on the following:

- An even x is heads
- An odd x is tails

Step One	Alice chooses an even/odd x (H or T)
Step Two	She sends $f(x)$ to Bob
Step Three	Bob calls H or T (odd or even), essentially guessing what kind of x produced the $f(x)$ he just got.
Step Four	Alice informs Bob whether he's right by sending him x
Step Five	Bob computes $f(x)$ and verifies that it agrees with the $f(x)$ Alice sent before.

Figure 5.1: Head/Tails Game protocol

Then we could play the game as in Table 5.2.2

This simple protocol shows that one-way functions can play an important role in securing content commitment without content disclosure.

A closely related kind of protocol is called a 0-knowledge protocol. In a zero-knowledge protocol A proves to B that A knows some bit of content without revealing to B what that content is. This is done by having B ask A a series of questions whose correct answers prove that A knows some piece of information (such as A's secret identity number) without revealing it.

This kind of protocol is very important in modern cryptography. For example, it is a central feature of digital cash schemes. There the basic requirement is that buyer and seller have a completely anonymous mechanism for digital funds transfer. But at the same time fraud prevention requires that there be some fallback means of identifying the buyer. Therefore digital cash protocols include some means by which the buyer commits to information revealing his or her identity, but that information only becomes available in cases of cheating (for example if an attempt is made to use the same funds twice).

5.2.3 The solution to several problems: Encryption

Much of cryptography involves encryption. An encryption has the following properties:

1. Plain text message. We call this **M** (for message)
2. Encoding algorithm. We call this **E**.
3. Decoding algorithm (perhaps even more important than 2!) We call this **D**.
4. Key. We call this **K**. In general the longer the key, the more secure the messages sent with it.

5. Cipher text. The result of encoding the message with the key. We call this **C**.

That's all.

Convention: $E(M,K) = C$.

Convention: $D(C,K) = M$.

A very example of a key would be the amount of shift in a shift cipher, as introduced in Section 2.2.4. Thus given that a shift cipher is being used the number 3 is enough information to uniquely decode every message using a shift-3 cipher. For an arbitrary cipher, the encoding matrix, a 26 X 26 permutation matrix (also discussed in Section 2.2.4), could function as a key. The general rule is, the greater the level of security required, the longer the key. No cipher is going to provide adequate security protection. Most keys for adequate encryption need to be represented by some number much too long for a human to remember

Another example of a key is the encryption key we used for Euler's Code, discussed in Section 4.4.3. There we noted that we actually needed two different keys, an encryption key and a decryption key. But since the decryption is directly computable from the encryption key by Euclid's Extended Algorithm, this was no great matter.

Why **a key**? Why not just a secret algorithm? so that:

$$E(M)=C$$

After all, knowing the algorithm does give an adversary a head start on the problem of cracking the code. If we just use a secret algorithm, keys would be unnecessary.

1. Short answer: Secrets always get out. Keys are easier to change than algorithms.
2. Longer answer: Make the algorithms public. Let the smartest mathematicians in the world amuse themselves by trying to find a weakness. Makes for the strongest possible encryption algorithms.

Encryption can solve two of the cryptographic problems we started with:

1. **Confidentiality:** Only those who know the key can read the message.

2. **Authenticity:** Bob wants to verify that a message came from Alice and Alice and Bob share a secret key known only to them. Bob challenges Alice to encrypt a random message, say, “2 is fun!” Only Alice can encrypt it correctly.

Integrity and nonrepudiation each call for a little something more, but secret keys can enter into solving these problems as well.

The problem with keys: Exchanging them.

Consider: The internet. Many to many communications among perfect strangers who cannot trust each other (but must in order for the American way to survive). For a community of n users,

$$\frac{n(n-1)}{1 * 2}$$

meetings are necessary to secure all pairwise communications.

The argument really applies to any distributed computing situation in which information must be divvied up any any number of equally important locations and cannot be gathered into a single authority. As long as the requirement is that information exchanged pairwise among any two of the peers cannot necessarily be shared among all of them, then some kind of public key solution will be attractive.

The classic case of this sort is E-commerce, where the secure information being exchanged is credit card numbers. The fact that there is typically one merchant and many customers is not the key security feature. The key security feature is that when a credit card number is transmitted the information must be sealed off from any participants to any other transaction or any eavesdroppers. Thus a single symmetric key for the merchant will not suffice. One key is required for each customer merchant pair, and potentially for each transaction.

5.2.4 Verifiable key exchange: A practical almost one-way almost-function

What we have called a one-way function allows one to combine publicly available information with some privately available formation to learn some content which is then private, for all practical purposes. Often the set of possible contents is large but not infinite. What a one-way function does, then, is to help choose from some intractably large set of alternatives. This can be done without the help of of an actual function.

Step One	Bob sends Alice 1,000,000 messages of the form <i>This is message number x. Use 128-bit key \mathbf{K}_x.</i> So each message gives Alice a different 128-bit key. He encrypts each of these messages using a randomly chosen 20-bit key [easily broken] whose identity doesn't matter.
Step Two	Alice randomly picks one of Bob's million messages and decrypts it using a brute force method based on the fact that it was encoded using a 20-bit key. She learns \mathbf{K}_x and uses it to encrypt a message m to Bob, sends the encrypted message, $E(\mathbf{K}_x, m)$, along with the message number x (not encrypted).
Step Three	Bob decrypts the message by looking up \mathbf{K}_x in message x .

Figure 5.2: Merkle secret key protocol

Figure 5.2.4 presents what seems to be the simplest idea, due to Ralph Merkle¹

Alice has to do some work to get her key. In fact for a key 20 bits long the task of breaking the code takes about 2^{20} (roughly a million) steps. But our evil eavesdropper Eve has to perform that same task for about 1,000,000 messages, so that's

$$2^{20} * 2^{20} = 2^{40}$$

steps, which is a **lot** more work.

This **might** be good enough. If Alice and Eve can both try 10K keys per second, Alice can do her decoding in about a minute on average, but Eve's decoding takes her about a year on average.

Note that there is no function here, one way or otherwise. What there is is a lot of values (Bob's 1,000,000 messages) and a random choice among them. The problem is determining what the result of that random choice is. Note that that determination isn't impossible, merely computationally difficult. Note that the amount of security depends on the difficulty of the computation for Eve. If

¹ The tale that is told among cryptographers about this is that Merkle took a course in Computer Security at Berkeley in 1974 [taught by Lance Hoffman]. He submitted a paper describing this idea; Hoffman couldn't understand it and Merkle dropped the course and eventually went on to become one of the people who developed the idea of public key cryptography. Retelling due to (Schneier 1996)

Eve works for the NSA she might have computational resources several orders of magnitude better than Alice's. Then the margin starts to dwindle away quickly.

The hassle: Bob has to do a lot of computing to generate 1,000,000 appropriate random messages, and he has to tie up a lot of bandwidth to send all of them to Alice. Alice has to do a lot of computing to get her key. Increasing the amount of security can be done by increasing the number of initial messages or increasing the size of the keys Bob sends (from 20 to, say, 25), thus increasing the amount of hassle.

Moral: A function that is **difficult** to undo would be better and could maybe secure a safer security margin without increasing the hassle.

5.2.5 Three true-to-life one-way functions

Here are three examples of one-way functions that might give us a better way to accomplish what Merkle's protocol was trying to accomplish in the last section.

1. Multiplication. Multiplication is easy. Factoring is hard. Factoring a number usually is taken to mean finding prime integers that divide it evenly. For very large numbers this is very hard. The idea is roughly this: Pick two primes p, q such that

$$pq = n \approx 10^{200}$$

For example:

$$\begin{aligned} p &= 8273488995874738949376376607 \\ q &= 15263784900967433245564811 \\ n &= 126284756413553050978360366248406608817836065776277 \end{aligned}$$

Multiplying p and q together to get n is easy (at least for a computer). Factoring n to find p and q is very hard.

2. Finding logarithms in modular arithmetic (for some large n). Raising a number a to a power x to get result r is easy in modular arithmetic, in fact, easier than in regular arithmetic. Finding what power a has to be raised to to get r is hard. This is called the *discrete logarithm* problem.
3. Taking roots in modular arithmetic. In general computing something like

$$\sqrt[5]{1} \pmod{11}$$

is quite difficult, while computing

$$5^5 \equiv 1 \pmod{11}$$

is quite easy. Obviously for small n $\sqrt[5]{x} \bmod n$ can be computed just by taking the 5-th power of all the numbers from 1 to $(n - 1)$ until we find x , but for large n this brute force approach is impractical.

Note that what counts here is not possibility or impossibility but the relative amount of computing.

5.2.6 Verifiable key exchange: A one-way function protocol

We want Alice and Bob to construct a key K they can use for their encryption. We want them to construct it by communications on an insecure channel. And of course we want K to be a secret.

Alice and Bob each contribute half the bits, we'll call Alice's bits x and Bob's bits y . Then we need some easy-to-compute function *construct-key* which computes K :

$$\text{construct-key}(x, y) = K$$

As before algorithms aren't assumed to be secret so *construct-key* is public.

Alice can't send x . That's insecure. So we assume a one-way function f as before.

Alice sends

$$f(x)$$

Bob sends

$$f(y)$$

The exact values of x and y are still secret because f is one-way.

What we now need is for Alice and Bob to both be able to construct the same key from what each has:

$$\begin{array}{ll} \text{Alice} & x \quad f(y) \\ \text{Bob} & y \quad f(x) \end{array}$$

So we need a version of *construct-key* which works even though one of its arguments has been passed through f :

$$\text{alt-construct-key}(f(x), y) = \text{alt-construct-key}(f(y), x) = \text{construct-key}(x, y) = K$$

Alice	x	$f(y)$	$\text{alt-construct-key}(f(y), x)$
Bob	y	$f(x)$	$\text{alt-construct-key}(f(x), y)$

For security we also want a kind of one-wayness for alt-construct-key . It should be impossible to construct K from $f(x)$ and $f(y)$; one must have either x or y . So there should be no easy-to-compute function break-key , such that:

$$\text{break-key}(f(x), f(y)) = K$$

In fact, we have already discussed functions of the sort we need in Section 5.2.5. These were multiplication in regular arithmetic and raising to a power in modular arithmetic; both have which inverses (factoring and discrete logarithms) which are hard to compute.

The reason these fit the bill is that we don't need to reach the ideal of functions that are impossible to run backwards. We just need a function that, given the complexity of the particular inputs we use, will take an impractical amount of computing time to compute, say 1000 years.

In fact, the protocol we have just described is the Diffie-Hellman protocol for key exchange. The one-way function used there is the discrete power function (inverse: (discrete log function). Details to be given in the next section.

5.2.7 Diffie-Hellman Protocol

First, Alice and Bob publically agree on on a large prime p and a generator g of Z_p^* (see Section 3.2.2). The protocol goes like this:

1. Alice chooses a large random number x and sends Bob:

$$X = g^x \pmod{p}$$

2. Bob chooses a large random number y and sends Alice:

$$Y = g^y \pmod{p}$$

3. Alice computes

$$K = Y^x \pmod{p}$$

4. Bob computes

$$K = X^y \pmod{p}$$

Note that Alice can do her key computation because she knows x , and Bob can do his because he knows y , but evil eavesdropper Eve doesn't know either of those numbers. Moreover Alice and Bob compute the same key because:

$$X^y = Y^x = g^{xy}$$

What Eve does know is g , p , X , and Y . But in order to compute the K she has to get back from X to x or from Y to y , and in order to do that she has to find what power g is raised to to give either X or Y . This is what is called the *discrete logarithm problem* and it is, as far as we know, very difficult to compute.

So raising to a power in modular arithmetic is our one way function. Recap- ping the original idea:

We needed Alice to send

$$f(x)$$

The f here was $g^x \pmod p$ In step 2, Bob sent

$$f(y) = g^y \pmod p$$

The exact values of x and y are still secret because f is one-way.

What we needed next was for Alice and Bob to both be able to construct the same key from what each has:

$$\begin{array}{ll} \text{Alice} & x \quad f(y) = Y \\ \text{Bob} & y \quad f(x) = X \end{array}$$

So we needed an alternative construct key function which works even though one of its arguments has gone through f :

$$\text{alt-construct-key}(Y, x) = \text{alt-construct-key}(X, y) = \text{construct-key}(x, y) = K$$

In the Diffie-Hellman protocol, alt-construct-key is just raising the last received message to a power $\pmod p$:

$$\begin{array}{ll} \text{alt-construct-key}(X, y) & X^y \pmod p \\ \text{alt-construct-key}(Y, x) & Y^x \pmod p \\ \text{construct-key}(x, y) & g^{xy} \pmod p \end{array}$$

And that was all there was to it, given that:

$$X^y = g^{xy} = Y^x = g^{yx} = g^{xy} \pmod p$$

5.2.8 Exercises

1. In the Diffie Hellman protocol, Alice and Bob start with a prime p . They then pick another number g which is a generator of Z_p^* . Why must g be a generator? HINT: The reasons don't have to do with making the math correct. They have to do with security.
2. The obvious way to test whether a candidate g is a generator of Z_p^* is just to compute g^i for all $0 < i \leq (p - 1)$. If $g^i = 1$ for any $i < (p - 1)$, then g is not a generator. But in fact one doesn't need to test all these i . It suffices to test all i such that $i \mid (p - 1)$. Why is this shortcut valid?

5.2.9 RSA protocol

The RSA protocol (named for its creators, Ron Rivest, Adi Shamir, and Leonard Adleman) goes as follows:

1. Choose two large random prime numbers p and q and multiply them together to give n

$$n = p \cdot q$$

2. Randomly choose an encryption key e such that e and $\phi(n)$ are relatively prime. Recall that $\phi(n)$, Euler's Totient function, is the number of integers relatively prime to n . It turns out that for an integer that is the product of two primes $\phi(n)$ is easy to compute *when you know the two primes*.
3. Use the extended Euclidean Algorithm to compute the inverse of $e \bmod \phi(n)$, d . That is, find d , $1 < d < \phi(n)$, such that:

$$e \cdot d \equiv 1 \pmod{\phi(n)}$$

d is the decryption key. Note that d and $\phi(n)$ are also relatively prime. Why?

4. The protocol will allow us to encrypt a message smaller than n . Longer messages will have to be broken up into blocks with each block encoded separately. Here is the encoding of a message m :

$$c = m^e \bmod n$$

5. The decoding of the message is as follows:

$$m = c^d \bmod n$$

That's all there is to it.

Why does it work? Clearly, it had better be the case that $c^d = m$. And it is. First let's just review how the pieces were put together:

$$c^d = (m^e)^d = m^{e \cdot d} \pmod n \quad (5.1)$$

In the following argument we will assume that the message m is relatively prime to n . Since n is the product of 2 primes, this takes care of just about all possible messages. The cases left over are the multiples of the two primes p and q from which n is derived. It turns out the protocol works for these cases as well, but the argument is slightly trickier, though similar (both arguments rely crucially on Euler's Theorem). To see the argument for this residue of cases, see the appendix, Section 6.2 We deal only with the core set of cases here.

Because e and d are inverses $\pmod{\phi(n)}$, there is some k such that:

$$e \cdot d = k \cdot \phi(n) + 1$$

Rewriting the last term in (5.1) we have:

$$m^{e \cdot d} = m^{k \cdot \phi(n) + 1} = m \cdot m^{k \cdot \phi(n)} = m \cdot (m^{\phi(n)})^k \pmod n \quad (5.2)$$

Now Euler's theorem applies in this case because we have assumed m is relatively prime to n . That is:

$$m^{\phi(n)} \equiv 1 \pmod n$$

so the last term in (5.2) reduces as follows:

$$m \cdot (m^{\phi(n)})^k \equiv m \cdot 1^k \equiv m \cdot 1 \equiv m \pmod n \quad (5.3)$$

Which is what we wanted to show.

So encryption and decryption work the way they should. The security of the algorithm is based on the fact that $\phi(n)$ is easy to compute for $n = p \cdot q$ if you know p and q , and very difficult otherwise.

In fact, $\phi(n) = (p - 1)(q - 1)$. This is fairly to easy to see.

Theorem 5.2.1. *Computation of $\phi(n)$*

Let $n = p \cdot q$, p, q primes. Then

$$\phi(n) = (p - 1)(q - 1)$$

Since p and q are prime, any integer not relatively prime to n must be a multiple of p or q . For the same reason, the smallest integer that is a multiple

of both p and q is pq , so there is no overlap between the set of multiples of p that are less than n and the set of multiples of q that are less than n . The multiples of p less than n are:

$$\{p, 2p, 3p, \dots, (q-1)p\}$$

There are $(q-1)$ such multiples. The set of multiples of q less than n is:

$$\{q, 2q, 3q, \dots, (p-1)q\}$$

There are $(p-1)$ such multiples. So the size of the set of integers less than n that are multiples of either p or q is:

$$(p-1) + (q-1) = p + q - 2$$

There are $p \cdot q - 1$ integers less than n . Therefore:

$$\phi(n) = (p \cdot q) - 1 - (p + q - 2) \quad (5.4)$$

$$= (p \cdot q) - p - q + 1 \quad (5.5)$$

$$= (p-1)(q-1) \quad (5.6)$$

Which is what we wanted to show.²

To compute $\phi(n)$, then all you have to do is factor n . But factoring large numbers is computationally very difficult; so the security of the algorithm will also be based on choosing very large p and q and getting an even larger n .

There is of course another way to retrieve m , without figuring out $\phi(n)$. This is to take the e th root of $m^e \pmod n$. But there are no known good algorithms for doing this either. The n th root problem in modular arithmetic appears to be at least as hard as if not equivalent to the factoring problem.

First let's try an example with small numbers. In fact, let's encrypt one letter, s ($= 19$ numerically) using $\pmod{55}$. We need to find $\phi(55)$. Well, it's $\phi(11) \cdot \phi(5) = 10 \cdot 4 = 40$. For our public key, we choose an exponent relatively prime to 40, 3. So to find our private key we need to find

$$3^{-1} \pmod{40}$$

²Note that Since p is a prime all integers less than p are relatively prime to p , so $\phi(p)$ is $(p-1)$, and similarly for q , $\phi(q)$ is $(q-1)$. Therefore in this case:

$$\phi(n) = \phi(p \cdot q) = \phi(p) \cdot \phi(q)$$

The appendix shows that this is true whenever p and q are relatively prime.

Using Euclid's Extended Algorithm we have:

$$\begin{array}{l}
 l_0, l_1 \quad \left[\begin{array}{ccc} l & \lambda & \mu \\ 40 & 1 & 0 \\ 3 & 0 & 1 \end{array} \right] \quad \begin{array}{l} l_0 = 40, l_1 = 3 \\ \lambda_0 = 1, \mu_0 = 0; \lambda_1 = 0, \mu_1 = 1; \\ \hline 40 = (1 \cdot 40) + (0 \cdot 3) \\ 3 = (0 \cdot 40) + (1 \cdot 3) \end{array} \\
 \hline
 l_2 \quad \left[\begin{array}{ccc} l & \lambda & \mu \\ 40 & 1 & 0 \\ 3 & 0 & 1 \\ \hline 1 & -13 & 1 \end{array} \right] \quad \begin{array}{l} q_2 = \lfloor 40/3 \rfloor = 13 \\ l_2 = 40 - (13 \cdot 3) = 1 \\ \lambda_2 = 0 - 1 \cdot 13 = -13 \\ \mu_2 = 1 - 0 \cdot 13 = 1 \\ \hline 1 = 1 \cdot 40 + (-13 \cdot 3) \end{array}
 \end{array}$$

So we have

$$3^{-1} = -13 \equiv 27 \pmod{40}$$

Verifying:

$$3 \cdot 27 = 81 \equiv 1 \pmod{40}$$

So we have public key 3 and private key 27. So let us encrypt a message one letter long choosing s (the 19th letter, which we represent with the number 19). Encoding then is:

$$19^3 \pmod{55} = 6859 \equiv 39 \pmod{55}$$

Now let's decrypt.

$$\begin{aligned}
 39^{27} &= 39^3 \cdot (39^8)^3 \pmod{55} \\
 &\equiv 29 \cdot (26)^3 \pmod{55} \\
 &\equiv 29 \cdot 31 \pmod{55} \\
 &\equiv 19 \pmod{55}
 \end{aligned}$$

Which, exactly as desired, is the representation for s .

As an example with larger numbers suppose Alice two large primes at random (There are good algorithms for testing the primality of a number; but that's a subject for another text):

$$\begin{aligned}
 \text{Let } p &= 2847\,893\,757\,848\,938\,511 \\
 q &= 92\,734\,928\,626\,327\,511
 \end{aligned}$$

Then

$$\begin{aligned}
 n &= p \cdot q \\
 &= 264\,099\,224\,369\,484\,956\,639\,974\,579\,586\,676\,121
 \end{aligned}$$

Alice computes $\phi(n) = (p-1)(q-1)$:

$$(p-1)(q-1) = 264\,099\,224\,369\,484\,953\,699\,345\,893\,111\,410\,100$$

and uses Euclid's algorithm to find some e relatively prime to $(p-1)(q-1)$:

$$\text{Let } e = 1009$$

This is Alice's encryption key, to be made public along with n . Then Alice computes the inverse of d , using Euclid's Extended Algorithm:

$$d = e^{-1} \bmod (p-1)(q-1)$$

$$5758\ 357\ 716\ 678\ 561\ 924\ 068\ 988\ 749\ 703\ 689$$

d , p and q are all secret.

Bob who wants to send Alice a peculiar message looks up e and n (this pair is called Alice's *public key*) on the internet³ He chooses the following peculiar message:

$$m = 33\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333$$

Then he encrypts his peculiar message as follows:

$$c = m^e \bmod n = 54\ 423\ 731\ 721\ 403\ 481\ 610\ 392\ 517\ 373\ 097\ 210$$

Alice gets the message and uses her *private key*, d , to decode it:

$$m = c^d = 33\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333\ 333$$

A significant and mathematically pretty property of the protocol is that it doesn't matter which key is used first and which second:

$$(m^e)^d = (m^d)^e = m$$

This means the same cryptographic tools can also be used to do *signatures* which assure authenticity of the message. Here's how that would work.

In addition to encoding a message to Alice using Alice's public key, Bob also encodes using his *secret key*. Using e_a , d_a , and n_a for Alice's public key, secret key, and modulus, and e_b , d_b , and n_b for Bob's public key, secret key, and modulus, Bob sends:

$$c = (m^{e_a} \bmod n_a)^{d_b} \bmod n_b$$

And Alice can decode this using two keys she knows, her secret key and Bob's public key (which she looks up on the internet):

$$m = (c^{e_b} \bmod n_b)^{d_a} \bmod n_a$$

If she gets something that looks like a real message that's good evidence that it was signed using Bob's secret key, because only such a message would make sense when decoded using Bob's public key. If she wants further assurance she can send Bob a specific challenge text, such as "Hello, world!", instructing him to encode it, and if the encoded version she receives back decodes to "Hello,

³ He might find it on her web-site (see for example <http://www.rohan.sdsu.edu/gawron>) or he might find it in a public key depository such as <http://pgpkeys.mit.edu/>.

world!” using Bob’s public key, then she has very good evidence indeed that it is really Bob on the other end of the channel.

In real life things are actually more complicated but this presents the main ideas of the original RSA paper and conveys the key breakthrough of public key cryptography. So this is as good a place as any to stop.

5.2.10 Exercises

1. Consider what has been called the *iterated encryption attack* on RSA. The basic idea is to continuously re-encrypt the encrypted message $c = m^e \pmod n$, using the same encryption key, until you have c again:

$$\begin{aligned} \text{Step 1} & \quad c^e \pmod n \\ \text{Step 2} & \quad (c^e)^e = c^{(e^2)} \pmod n \\ \text{Step 3} & \quad (c^{e^2})^e = c^{(e^3)} \pmod n \\ & \quad \vdots \\ \text{Step } i & \quad (c^{e^{i-1}})^e = c^{(e^i)} \pmod n \end{aligned}$$

Now in some cases you will eventually reach a step k such that:

$$\text{Step } k \quad c = c^{(e^k)} \pmod n$$

At this point we can stop encrypting. The attack is complete. The original message can be recovered. Your mission for this problem is to determine what m is given that you have reached step k successfully.

The more interesting question of when this can happen and how the iterated encryption attack can be blocked is a matter for another text.

5.3 Practice versus Textbook

The two public key protocols been presented in this book are the RSA and Diffie-Hellman protocols. For quite a while these were the two most important protocols in the area of public key cryptography; arguably they still are though competitors exist. The important point to make here is that a number of simplifications have been made in the presentations in this chapter. To start with the versions presented are in both cases textbook versions. A number of practical issues have been finessed or ignored entirely. This is less of a problem than it might be because the goal of this book is to explore the relationships between the applications and the mathematical ideas

Two basic kinds of simplification have been at work, computational and security-related. An example of a computational simplification is that we haven't talked at all about how to choose primes, which is critical for RSA key generation. In fact, it is far from trivial to choose primes efficiently and current state-of-the-art methods aren't guaranteed to return primes; they are only guaranteed to almost certainly return primes, which seems to be good enough. An example of a security-related simplification is that we haven't talked about how advances in factoring have raised issues about which primes are *safe* to use for encryption. Basically, with the advent of something called the elliptic curve factoring algorithm, primes have to be *large enough* to be safe, where large enough is larger than it used to be.

Another example of security-related simplification, inherited from the original RSA paper, is our description of digital signatures. In fact, signing an entire message with one's private key d is a very bad idea. It is open to an attack in which Eve sends a previous encrypted message from Alice to Bob and Bob signs it and returns it (perhaps the result of an automatic email handling system). Basically when Bob signs that message he applies decryption key d to encrypted message m^e and

$$(m^e)^d = m$$

The result is that "signed message" he sends Eve is the original plain text m . Moreover it isn't enough to just be on the lookout for previously encrypted ciphertexts. There are numerous undoable transformations that can be performed on m^e so as not to make it identical to a previous message. There are two ways to prevent this attack. First, never sign an entire message; rather sign a *digest of a message*, where a digest is a number returned by something called a hash function, which is sensitive to the smallest changes in the message and whose variation for any given change is very hard to predict. A hash function is thus another one-way function. Thus if a signed digest is authenticated, one can be sure both that the message is really from the purported sender and that it has not been altered. The second, equally important practice is never to use the same publickey/secret key pair for both signature and content encryption.

Another example of a computationally related simplification is our description of message encryption in RSA. In practice, the RSA public key is almost never used to encrypt the entire message m , because public key encryption is so much less efficient than old style symmetric key encryption (the price would actually be paid at decryption time, because the decryption key is usually the bigger one). Instead what is done is to encrypt the entire message using some standard symmetric key encryption algorithm such as AES or DES and then to encrypt the AES or DES *session key* using RSA. What is encrypted using RSA is just a 256-bit key and thus decrypts much more quickly.

The original advertised difference between RSA and Diffie-Hellman was that Diffie-Hellman was a protocol for key exchange and that RSA was a protocol

for message encryption and digital signatures. Looking at the textbook versions, the two protocols ought not to be in competition. But in practice, for the efficiency reasons just cited, RSA becomes a key-exchange protocol too, and thus, to achieve secure efficient communication, both protocols need to be supplemented with ordinary symmetric encryption. The underlying mathematics remains quite distinct as presented here, but the two protocols often fill quite similar roles in their realizations in actual systems. It is thus not that surprising that, as a matter of historical fact, the two companies holding the now-expired RSA and Diffie-Hellman patents, with different algorithms with different purposes, with their respective MIT and Stanford roots, were ultimately joined under a single corporate umbrella PKP (Public Key Partners).

Real systems are also built to ensure that failures of one module have as little effect as possible on other modules. For example knowledge of pairs of messages and signatures can sometimes be used to predict the signatures on other messages. Thus it increases security to insulate plain-text/encrypted text pairs as much as possible from discovery. In this spirit, Ferguson and Schneier (Ferguson and Schneier 2003) recommend that the actual symmetric session key K in RSA practice be the result of applying a hash function to the key K' that is encrypted using RSA. Thus, if the session key K is compromised, and the message that transmitted it is known, the sequence K' that gave rise to K is still unknown (since hash functions are one-way). Let K' be the original randomly chosen key and K'^e be its RSA encoding, which is transmitted and made public. Now let h be the hash function. Then:

$$K = h(K')$$

So the computation for getting from the transmitted message K'^e to K is:

$$K = h((K'^e)^d)$$

Considering that all we are talking about here is the session key part of the protocol, before a single word of content has been transmitted, things are already getting pretty complicated.

Another huge area of practical concern which has gone undiscussed here is how public keys are to be made available. The whole point of public keys is — well, that they are public. Accessibility is what makes public keys useful, but it is also what makes them vulnerable. How is Alice to know when she looks up Bob's key that the key she finds is authentic? For example, if Alice downloads the public key from Bob's web-site, how can she verify that the site she was linked to at download time was really Bob's, and not some spoof site prepared by Eve? There are a variety of answers to such questions and a useful discussion is beyond the scope of this book., but two general points are of interest. First, some trusted entity is needed, either a central key authority or a web of trusted users, from which key information can be obtained and/or verified. Second, and independently of the first point, there needs to be some

way of verifying keys over a secure channel. A typical choice is that keys are hashed, producing a manageable sized “fingerprint” which Bob can read over the phone to Alice; Alice then matches what she hears over the phone to the fingerprint she generates from Bob’s downloaded key, and is reassured that that key is authentic. In the end, though, public key infrastructure (PKI) raises a whole host of new issues that are still being worked out. For instance, when Alice’s private key is compromised the situation is a good deal more serious than it is when a symmetric key is compromised, because there is a whole community of users who still believe they can send Alice secure communications using her public key.

The point of this section is to underscore that the theory sketched here is a long way from being practice. Yet as these examples also suggest, the mathematical ideas discussed here, rooted deeply in the purest branches of pure mathematics, still play a central role in making the final system secure.

Readers interested in plumbing the practical depths should turn to other texts, beginning with Schneier’s excellent Schneier (1996). Far more enmired in the nitty gritty and benefiting from several years of real world experience is Ferguson and Schneier Ferguson and Schneier (2003). For a less technical discussion that still airs out a lot of the practical issues, see Mel and Baker Mel and Baker (2001).

Chapter 6

Appendix

6.1 Computing Euler's Totient Function

Euler's Totient function ϕ is defined as follows:

$\phi(n)$ is the number of integers y , $1 < y \leq n$ such that $y \perp n$

In this section we will prove the following theorem:

Theorem 6.1.1. *Product Theorem for Euler's Totient Function*

$$\text{If } p \perp q, \phi(p \cdot q) = \phi(p) \cdot \phi(q)$$

To prove this, we arrange the numbers from 1 to $p \cdot q$ in the following array:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & \dots & q \\ q+1 & q+2 & q+3 & q+4 & \dots & 2q \\ 2q+1 & 2q+2 & 2q+3 & 2q+4 & \dots & 3q \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (p-1)q+1 & (p-1)q+2 & (p-1)q+3 & (p-1)q+4 & \dots & pq \end{array} \quad (6.1)$$

We first show that a number x is relatively prime to $p \cdot q$ if and only if it is relatively prime to both p and q . This reduces our task to one of proving facts about p and q . We then prove two lemmas about columns (or *column lemmas*) in array (6.1). We show that there are exactly $\phi(q)$ columns that contain numbers relatively prime to q , and that all the numbers in those columns are relatively prime to q . We then show that every column contains exactly $\phi(p)$ numbers

relatively prime to p , so of those $\phi(q)$ columns that contain numbers relatively prime to q , each contains exactly $\phi(p)$ numbers also relatively prime to p , so there are $\phi(q) \cdot \phi(p)$ numbers that satisfy our criteria. And we are done.

Definition 6.1.1. \nmid We use $x \nmid q$ to mean that x is not relatively prime to q .

Lemma 6.1.1. $x \perp pq$ if and only if $x \perp p$ and $x \perp q$

Any number that divides p divides $p \cdot q$, so if, for some x , $x \nmid p$ then $x \nmid p \cdot q$. From which it follows that if $x \perp p \cdot q$, then $x \perp p$. Arguing similarly for q , we have if $x \perp p \cdot q$, then $x \perp q$. In other words, if $x \perp p \cdot q$, then $x \perp p$ and $x \perp q$.

For the other direction, any divisor of $p \cdot q$ must be composed of factors of p and/or q , so if $x \nmid (p \cdot q)$ then either $x \nmid p$ or $x \nmid q$. Thus if $x \perp p$ and $x \perp q$ then $x \perp (p \cdot q)$. This completes the proof of the lemma.

We now turn to our column lemmas. We can rewrite the number x in the i th row and j th column as:

$$(i-1)q + j$$

The first lemma is about how a number that can be written this way can be relatively prime to q :

Lemma 6.1.2. Relative primes of q

Let x be the number in the i th row and j column of (6.1):

$$x = (i-1)q + j \tag{6.2}$$

Then $x \perp q$ if and only if $x \perp j$

Proof: Note first of all that if q and j have a factor greater than 1 in common, then x also has that factor. If q and j have a factor in common, then there exists f, r_1, r_2 such that:

$$x = (i-1)f \cdot r_1 + f \cdot r_2 = f \cdot ((i-1)r_1 + r_2)$$

So x also has that factor. So we have:

$$\text{If } j \nmid q \text{ then } x \nmid q$$

Or equivalently:

$$\text{If } x \perp q \text{ then } j \perp q$$

Going the other direction: Suppose $j \perp q$. We now show $x \perp q$.

If $x \nmid q$, then there is some f, r_1, r_2

$$x = f \cdot r_1 \text{ and } q = f \cdot r_2$$

Therefore (6.2) may be rewritten as:

$$\begin{aligned} f \cdot r_1 &= (i-1)f \cdot r_2 + j \\ f \cdot r_1 - f \cdot r_2 &= j \\ f(r_1 - r_2) &= j \end{aligned}$$

Therefore $j \not\perp q$

So we have:

$$\text{If } x \not\perp q \text{ then } j \not\perp q$$

Or equivalently:

$$\text{If } j \perp q \text{ then } x \perp q$$

Which is what we wanted to show.

From this lemma it follows that that each column with a j relatively prime to q consists entirely of integers relatively prime to q . There are $\phi(q)$ such j with $1 \leq j < q$. Therefore.

Column lemma 1: There are $\phi(q)$ columns in (6.1) that consist entirely of numbers relatively prime to q .

This is the first of our two column lemmas. The second of our column lemmas is:

Column lemma 2: Every column in (6.1) has $\phi(p)$ numbers that are relatively prime to p .

It will be easiest to get to this lemma by way of a theorem that concerns the notion of a complete residue system of integers mod n (this concept was defined in definition 3.2.1, repeated here).

Definition 6.1.2. *Complete Residue System mod n*

A Complete residue system mod n is a set of integers such that

1. *Minimality: For any integer x there is $y \in R$ such that $x \equiv y$.*
2. *Completeness If $x, y \in R$, and $x \equiv y$ then $x = y$*

The following theorem is the multiplicative analogue of the additive theorem (3.2.1) of complete residue systems

Lemma 6.1.3. *Product Theorem for Complete Residue Systems*

If R is a complete residue system mod n and $p \perp n$, then

$$p \cdot R = \{p \cdot x \mid x \in R\}$$

is a complete residue system mod n .

Let $x \cdot p, y \cdot p \in p \cdot R$, with $x, y \in R$: And let:

$$x \cdot p \equiv y \cdot p \pmod{n} \tag{6.3}$$

Then, since $p \perp n$, p has an inverse mod n , and:

$$\begin{aligned} x \cdot p \cdot p^{-1} &\equiv y \cdot p \cdot p^{-1} \pmod{n} \\ x &\equiv y \pmod{n} \end{aligned}$$

This implication runs trivially in the opposite direction as well. Therefore $p \cdot R$ has exactly as many equivalence classes as R . And since R , by assumption, was already complete and minimal, we are done.

This lemma taken together with Theorem 3.2.1 immediately leads to the following theorem:

Theorem 6.1.2. *Linear Combination Theorem for Complete Residue Systems*

If R is a complete residue system mod n , $p \perp n$ and j is an integer, then

$$\{p \cdot x + j \mid x \in R\} \tag{6.4}$$

is a complete residue system mod n .

By Lemma 6.1.3, R' , where

$$R' = \{p \cdot x \mid x \in R\}$$

is a complete residue system mod n . Therefore by Theorem 3.2.1, R'' is, where

$$R'' = \{x + j \mid x \in R\}$$

But R'' is just the set in (6.4).

It follows from Theorem 6.1.2 that each column of (6.1) is a complete residue system mod p . First R is a complete residue system mod p :

$$R = \{0, 1, 2, 3, 4, \dots, (p-1)\}$$

6.2. RSA ALGORITHM: WHAT IF M IS NOT RELATIVELY PRIME TO N 87

The j th column of (6.1), then, is given by:

$$q \cdot \mathbb{R} + j = \{q \cdot i + j \mid i \in \mathbb{R}\}$$

Since $p \perp q$, Theorem 6.1.2 applies and $q \cdot \mathbb{R} + j$ is also a complete residue system mod p .

Any complete residue system mod p has exactly the same modular arithmetic properties as \mathbb{R} . In particular it has the same number of integers with inverses, $\phi(p)$ (recall that any integer with an inverse must be relatively prime to p , by Theorem 3.4.1).

Thus we have now established both our column lemmas. There are $\phi(q)$ columns consisting entirely of numbers relatively prime to q and for any column, there are $\phi(p)$ integers in it that are relatively prime to p . Since any integer relatively prime to $p \cdot q$ must be relatively prime to both p and q , there are $\phi(q)\phi(p)$ such integers. Which was what needed to be proved.

Example: This is one of those theorems whose proof is informative enough to make very clear claims about how something is computed. Consider the set of integers relatively prime to $36(= 9 \cdot 4)$, set in boldface in the following array:

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36

$\phi(4) = 2$ and $\phi(9) = 6$, $4 \perp 9$, so we expect $\phi(36)$ to be 12, and it is. And we expect those relative primes to appear in a certain pattern and they do. Each column has exactly $\phi(4)$ ($= 2$) numbers relatively prime to 4 (the 2 odd numbers), and there are $\phi(9)$ ($= 6$) columns all of whose numbers are relatively prime to 9. That gives us a total of 12 numbers relatively prime to 36.

6.2 RSA Algorithm: What if m is not relatively prime to n

We have for purposes of the RSA algorithm that $n = p \cdot q$, p, q prime. In the text we considered the case where m , the message, is relatively prime to n . We now consider the case where it is not. What we need to show to verify RSA is that:

$$m^{ed} \equiv m \pmod{n}$$

in the case where m and n are not relatively prime.

Given that n is the product of two primes, the only way m can not be relatively prime to n is if m is a multiple of one of the two primes. We will take the case where $p \mid m$. The other is symmetric. Note that we cannot also have that $q \mid m$ because the smallest number x such that $p \mid x$ and $q \mid x$ is $p \cdot q$ and we assume $m < p \cdot q$. Since q is a prime it is relatively prime to all numbers except its multiples and we have just shown m is not a multiple. So $q \perp m$.

So we have $p \mid m$ and $q \perp m$. It is now easy to show:

$$m^{ed} \equiv m \pmod{q} \quad (6.5)$$

$$m^{ed} \equiv 0 \pmod{p} \quad (6.6)$$

First we show (6.5). From Theorem 5.2.1, we know $\phi(n) = (p-1)(q-1)$. Recall the e and d are inverses $\pmod{\phi(n)}$. Therefore $e \cdot d \equiv 1 \pmod{\phi(n)}$ and we may represent $e \cdot d$ as:

$$k(p-1)(q-1) + 1$$

We then have

$$m^{e \cdot d} = m^{1+(p-1)(q-1)k} \pmod{q} \quad (6.7)$$

$$= m \cdot m^{(p-1)(q-1)k} \pmod{q} \quad (6.8)$$

$$= m \cdot (m^{(q-1)})^{k(p-1)} \pmod{q} \quad (6.9)$$

$$\equiv m \cdot (1)^{k(p-1)} = m \pmod{q} \quad (6.10)$$

Step (6.10) invokes Euler's Theorem, which is justified, since $m \perp q$, q is prime, and $\phi(q) = (q-1)$.

Next we show (6.6), which is trivial. By assumption $m \mid p$. Therefore we have:

$$m \equiv 0 \pmod{p} \quad (6.11)$$

$$m^{ed} \equiv 0 \pmod{p} \quad (6.12)$$

We repeat here the properties m^{ed} has in (6.5) and (6.6):

$$m^{ed} \equiv m \pmod{q}$$

$$m^{ed} \equiv 0 \pmod{p}$$

Note that one number that has the same two properties is m :

$$m \equiv m \pmod{q}$$

$$m \equiv 0 \pmod{p} \quad \text{since } m \mid p$$

We next show that for any y such that:

$$y \equiv m \pmod{q} \quad (6.13)$$

$$y \equiv 0 \pmod{p},$$

6.2. RSA ALGORITHM: WHAT IF M IS NOT RELATIVELY PRIME TO N 89

$y \equiv m \pmod{p \cdot q}$. We have:

- (a) $y \equiv m \pmod{p}$ Assumption (6.14)
- (b) $m \equiv m \pmod{p}$
- (c) $(y - m) \equiv 0 \pmod{p}$ Subtracting (b) from (a)
- (d) $y \equiv 0 \pmod{q}$ Assumption
- (e) $m \equiv 0 \pmod{q}$ (6.11)
- (f) $(y - m) \equiv 0 \pmod{q}$ Subtracting (e) from (d)

Therefore we have $p \mid (y - m)$ and $q \mid (y - m)$ from Steps (6.14c) and (6.14f). Since $p \perp q$, it follows that $(p \cdot q) \mid (y - m)$. Therefore:

$$\begin{aligned} (y - m) &\equiv 0 \pmod{(p \cdot q)} \\ y &\equiv m \pmod{(p \cdot q)} \end{aligned}$$

We have now shown that any y that has the two properties in (6.13) is congruent to $m \pmod{p \cdot q}$. Since m^{ed} is one such y , it follows that:

$$\begin{aligned} m^{ed} &\equiv m \pmod{(p \cdot q)} \\ m^{ed} &\equiv m \pmod{n} \end{aligned}$$

Which is what was to be shown.¹

¹ Note: The proof given here is actually a special case of what is known as the Chinese Remainder Theorem, for those following along in other texts. The Theorem states that if $m \perp n$, then the pair of equations:

$$\begin{aligned} x &\equiv a \pmod{m} \\ x &\equiv b \pmod{n} \end{aligned}$$

has a solution and that any two solutions are congruent modulo $m \cdot n$. The theorem can be formulated even more generally using the notions GCD and least common multiple, but that is a matter for another text.

Bibliography

Ferguson, Niels, and Bruce Schneier. 2003. *Practical Cryptography*. New York: John Wiley & Sons.

Mel, H X, and Doris Baker. 2001. *Cryptography Decrypted*. New York: Addison-Wesley.

Schneier, Bruce. 1996. *Applied Cryptography*. New York: John Wiley & Sons.