1

Introduction to Ciphers and Substitution

1.1 ALICE AND BOB AND CARL AND JULIUS: TERMINOLOGY AND CAESAR CIPHER

People have been trying to hide the content of written messages almost as long as writing itself has existed and have developed a multitude of different methods of doing it. And almost as soon as people started trying to hide their messages, scholars started trying to classify and describe these methods. Unfortunately, this means that I've got to hit you straight up with a bunch of terminology. Even worse, a lot of words that are used interchangeably in ordinary conversation have specific meanings to experts in the field. It's not really hard to get the hang of what's what, though.

As our first example, people who study secret messages often use the terms **code** and **cipher** to mean two different things. David Kahn, author of perhaps the definitive account of the history of cryptography, said it about as well as anyone could: "A code consists of thousands of words, phrases, letters, and syllables with the codewords or codenumbers . . . that replace these plaintext elements . . . . In ciphers, on the other hand, the basic unit is the letter, sometimes the letter-pair . . . , very rarely larger groups of letters . . . ." A third method of sending secret messages, **steganography**, consists of concealing the very existence of the message, for example, through the use of invisible ink. In this book we will concentrate on ciphers as they are generally the most interesting mathematically, although examples of the other methods may come up from time to time.

A few more terms will be helpful before we get started. The study of how to send secret messages by codes and ciphers is called **cryptography**, whereas the study of how to read such secret messages without

permission is called **cryptanalysis**, or **codebreaking**. Together, the two fields make up the field of **cryptology**. (Sometimes cryptography is also used for the two fields combined, but we will try to keep the terms separate.)

It's become customary when talking about cryptology to talk about Alice, who wants to send a message to Bob. We're going to start with Julius, though. That's **Julius Caesar**, who in addition to being *dictator perpetuus* of Rome was also a military genius, a writer, and . . . a cryptographer.

Caesar probably wasn't the original inventor of what we now call the **Caesar cipher**, but he certainly made it popular. The Roman historian Suetonius describes the cipher:

> There are also letters of his [Caesar's] to Cicero, as well as to his intimates on private affairs, and in the latter, if he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others.

In other words, whenever Alice wants to send a message, she first writes out the **plaintext**, or the text of the message in ordinary language. She is going to **encipher** this message, or put it into secret form using a cipher, and the result will be the **ciphertext** of the message. To put it into code would be to **encode** it, and the term **encrypt** can be used for either. For every a in the plaintext, Alice substitutes D in the ciphertext, for every b, she substitutes E, and so on. Each letter is shifted 3 letters down the alphabet. That's perfectly straightforward. The interesting part happens when Alice gets to the end of the alphabet and runs out of letters. The letter w becomes Z, so where does the letter x go? It wraps around, to A! The letter y becomes B and z becomes C. For example, the message "and you too, Brutus" becomes

plaintext:	a	n	d	y	o	u	t	o	o	b	r	u	t	u	s
ciphertext:	D	Q	G	B	R	X	W	R	R	E	U	X	W	X	V

This would be the message Alice sends to Bob.

You have actually used this "wraparound" idea in daily life since you were a child. What's 3 hours after 1:00? It's 4:00. Three hours after 2:00 is 5:00. What's 3 hours after 10:00? It's 1:00. You wrapped around.

It was around 1800 CE when **Carl Friedrich Gauss** codified this wraparound idea formally. It's now called **modular arithmetic**, and the wraparound number is called the **modulus**. A mathematician would write our wraparound clock example like this:

$$10 + 3 \equiv 1 \pmod{12}$$

and read it as "ten plus three is one modulo twelve."

But what about Caesar cipher? We can represent it using modular arithmetic if we are willing to change our letters into numbers. Instead of a think of the number 1, instead of b think of the number 2, and so on. This changing of letters to numbers is not really considered part of the secret cipher. It's a pretty obvious idea to those of us in the digital age, and Alice shouldn't really expect to keep it a secret. Only the operations that we do on the numbers are considered secret.

Now our modulus is 26 and our Caesar cipher looks like this.

plaintext	number	plus 3	ciphertext
a	1	4	D
b	2	5	E
⋮	⋮	⋮	⋮
x	24	1	A
y	25	2	B
z	26	3	C

Remember that the "plus 3" wraps around at 26.

To **decipher** the message, or take it from ciphertext to plaintext, Bob shifts three letters in the opposite direction, left. This time, he has to wrap around when he goes past a, or in terms of numbers, when he goes past 1. 0 wraps to 26, −1 wraps to 25, and so on. In the form we used earlier, that looks like the following.

ciphertext	number	minus 3	plaintext
A	1	24	x
B	2	25	y
C	3	26	z
⋮	⋮	⋮	⋮
Y	25	22	v
Z	26	23	w

1.2 THE KEY TO THE MATTER: GENERALIZING THE CAESAR CIPHER

From Caesar's point of view, he had a pretty secure cipher. After all, most of the people who might intercept one of his messages couldn't even read, much less analyze a cipher. However, from a modern cryptologic point of view it has a major drawback—once you have figured out that someone is using Caesar cipher, you know everything about the system. There's no **key**, or extra piece of information, that lets you vary the cipher. This is considered to be a very bad thing.

Stop to think about that a moment. What's the big deal? Your cipher is either secret or it isn't, right? That was the view in Caesar's time and for many centuries afterward. But in 1883, **Auguste Kerckhoffs** published a revolutionary essay, in which he stated, "The system must not require secrecy and can be stolen by the enemy without causing trouble." Amazing! How can having your system stolen not cause trouble?

Kerckhoffs went on to point out that it is just too easy for Eve the Eavesdropper to find out what system Alice and Bob are using. In Kerckhoffs' time, like Caesar's, cryptography was used mostly by militaries and governments, so Kerckhoffs was thinking about the information that an enemy might get through bribing or capturing a member of Alice or Bob's staff. These are still valid concerns in many situations today, and we can add to them the possibilities of Eve tapping phone lines, installing spyware on computers, and plain lucky guessing.

On the other hand, if Alice and Bob have a system that requires a key to encipher and decipher, then things aren't so bad. If Eve finds out what general system is being used, she still can't easily read any messages. Attempting to read a message without the key and/or determining the key used for a message is called **cryptanalyzing** the message or cipher or, more colloquially, **breaking** it. And even if she manages to

find out Alice and Bob's key, all is not lost. If Alice and Bob are smart, they are changing the key regularly. Since the basic system is the same, this isn't too hard, and then even if Eve gets the key to some of the messages, she won't be able to read all of them.

So we need to find a way to make small changes to Caesar cipher, depending on the value of some key. A logical place to start would be to ask why Alice is shifting her plaintext 3 places and not some other number? There is no particular reason; perhaps Caesar was just fond of the number 3. His successor, Augustus, used a similar system but shifted each letter only 1 place to the right. The "rot13" ("rot" stands for rotate) cipher shifts each letter forward by 13 places, wrapping around when you get to the end. This cipher is often used on the Internet to hide the punchlines of jokes or things that some people might find offensive. The general idea of shifting by k letters (or adding k modulo 26) is called a **shift cipher**, or **additive cipher**, with a key of k. For example, consider a shift cipher with a key of 21. Then Caesar's message would be:

plaintext:	a	n	d	y	o	u	t	o	o	b	r	u	t	u	s
numbers:	1	14	4	25	15	21	20	15	15	2	18	21	20	21	19
plus 21:	22	9	25	20	10	16	15	10	10	23	13	16	15	16	14
ciphertext:	V	I	Y	T	J	P	O	J	J	W	M	P	O	P	N

How many different keys are there? Shifting by 0 letters is probably not a good idea, but you could do it. Shifting by 26 letters is the same as shifting by 0 letters—or, in other words, 26 is the same as 0 modulo 26. Shifting by 27 letters is the same as shifting by 1 letter, and so on. So there are 26 ways of shifting that actually give you different results, or 26 keys. Note that this includes 0, the "stupid key," which doesn't do anything to the message. The technical term for when a cipher doesn't do anything is the **trivial cipher**. Suppose Alice sends Bob a message using a shift cipher and Eve intercepts it. Even if Eve has somehow learned that Alice and Bob are using a shift cipher, she still has to try 26 different keys in order to decipher the message. That's not a large number, but it's better than Caesar cipher.

Can we add some more keys? What about shifting our letters left instead of right? Unfortunately, that doesn't help. Suppose we shift our plaintext 1 letter to the left and wrap around the other direction.

plaintext:	a	n	d	y	o	u	t	o	o	b	r	u	t	u	s
numbers:	1	14	4	25	15	21	20	15	15	2	18	21	20	21	19
minus 1:	0	13	3	24	14	20	19	14	14	1	17	20	19	20	18
ciphertext:	Z	M	C	X	N	T	S	N	N	A	Q	T	S	T	R

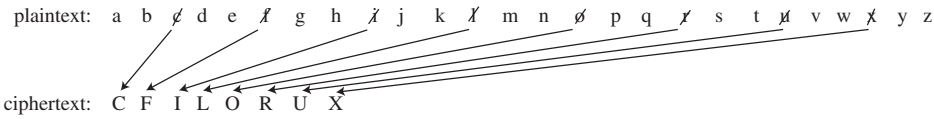
Note that since 0 is the same as 26 modulo 26, we can assign them both to the ciphertext letter "Z" interchangeably. If you think about it, you'll see that shifting 1 letter to the left is the same as shifting 25 letters to the right. Or in terms of modular arithmetic, you can think of left shifts as negative, so we are saying -1 is the same as 25 modulo 26. So left shifts don't help.

1.3 MULTIPLICATIVE CIPHERS

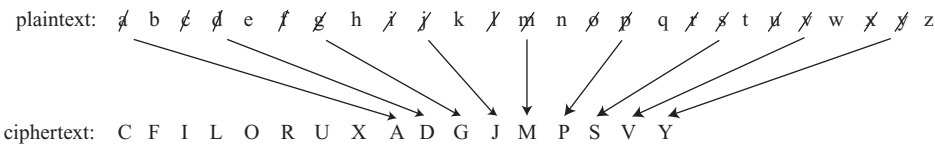
Let's look at a different type of cipher for some inspiration. This is called the **decimation method** of constructing a cipher. We need to pick a key, say 3. We start by writing out our plaintext alphabet.

plaintext: a b c d e f g h i j k l m n o p q r s t u v w x y z

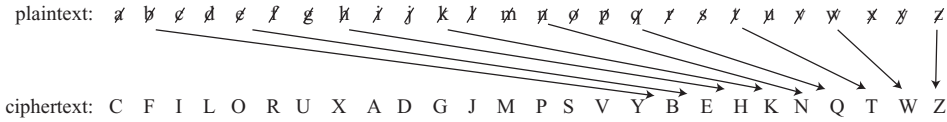
Then we count off every third letter, crossing each out (or "decimating" it) and writing each such letter as our ciphertext alphabet.



When you get to the end, "wrap around" to the beginning. In this case, cross out the "a" and keep going.



Finally, wrap around to the "b" and finish up:



So our final translation of plaintext to ciphertext is

plaintext:	a	b	c	d	e	f	g	h	i	j	k	l	m
ciphertext:	C	F	I	L	O	R	U	X	A	D	G	J	M
plaintext:	n	o	p	q	r	s	t	u	v	w	x	y	z
ciphertext:	P	S	V	Y	B	E	H	K	N	Q	T	W	Z

Okay, now let's try to look at this like a mathematician. How can we describe the decimation method in terms of modular arithmetic? Well, we should translate our numbers into letters, of course.

plaintext:	a	b	c	d	e	f	g	h	i	j	···	y	z
numbers:	1	2	3	4	5	6	7	8	9	10	···	25	26
some operation?:	3	6	9	12	15	18	21	24	1	4	···	23	26
ciphertext:	C	F	I	L	O	R	U	X	A	D	···	W	Z

Very interesting! For the first eight letters, all we have to do is multiply the number corresponding to the plaintext by 3 (the key) and we get the ciphertext. For the letter i this doesn't quite work, because 9 times 3 is 27—but 27 is the same as 1 modulo 26, which corresponds correctly to our ciphertext letter A.

Apparently there was nothing much special about the addition part of our additive cipher. Instead of adding 3 to each plaintext number, we can multiply by 3 instead, wrapping around when we get to 26. This makes sense from the "clock arithmetic" point of view also: Start at midnight. Three times 3 hours later is 9:00. Three times 4 hours later is 12:00. Three times 5 hours later is 3:00, and so on. Our new **multiplicative cipher** with key 3 looks like this:

plaintext	number	times 3	ciphertext
a	1	3	C
b	2	6	F
⋮	⋮	⋮	⋮
y	25	23	W
z	26	26	Z

If we want to encipher the message "be fruitful and multiply," it would look like this:

plaintext:	b	e	f	r	u	i	t	f	u	l	a	n	d
numbers:	2	5	6	18	21	9	20	6	21	12	1	14	4
times 3:	6	15	18	2	11	1	8	18	11	10	3	16	12
ciphertext:	F	O	R	B	K	A	H	R	K	J	C	P	L

plaintext:	m	u	l	t	i	p	l	y
numbers:	13	21	12	20	9	16	12	25
times 3:	13	11	10	8	1	22	10	23
ciphertext:	M	K	J	H	A	V	J	W

Incidentally, it's often useful to have a faster way of dealing with the wraparound than subtracting 26 over and over again. Luckily, you already know one—it's division with remainder, just like you learned in grade school. Only now, once we have seen how many 26s go into the number, we are going to throw all the 26s away and just keep the remainder. For example, to encipher the last letter of the preceding example, I multiplied 25 by 3 to get 75. Then I divided 75 by 26:

$$\begin{array}{r} 2 \\ \hline 26\,\big)\;\;75 \\ -52 \\ \hline 23 \end{array}$$

The quotient is 2, which I can throw away, and the remainder is 23, which is the number I need for my ciphertext. Another way of thinking about it is that the division with remainder shows that $75 = 2 \times 26 + 23$; that is, 75 is twice 26 with 23 left over. But 26 is the same as 0 modulo 26, so 75 is the same as $2 \times 0 + 23 = 23$ modulo 26.

How many keys does the multiplicative cipher have? At first glance, you might expect 26 again, including one stupid key. But hold on a moment—multiplying by 26 modulo 26 is the same as multiplying by 0. And multiplying by 0 is *bad*. Not just stupid, but bad. A multiplicative cipher with a key of 0 looks like this:

plaintext	number	times 0	ciphertext
a	1	0	Z
b	2	0	Z
⋮	⋮	⋮	⋮
y	25	0	Z
z	26	0	Z

So if we encrypt a message with this cipher, it comes out as

plaintext:	a	r	e	a	l	l	y	b	a	d	k	e	y
numbers:	1	18	5	1	12	12	25	2	1	4	11	5	25
times 0:	0	0	0	0	0	0	0	0	0	0	0	0	0
ciphertext:	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

There's no way on earth to decrypt that! So we can't use that key.

Are there any other keys we can't use? Think about multiplying by 2—we know that any number multiplied by 2 is even. A multiplicative cipher with a key of 2 looks like this:

plaintext	number	times 2	ciphertext
a	1	2	B
b	2	4	D
⋮	⋮	⋮	⋮
l	12	24	X
m	13	26	Z
n	14	2	B
o	15	4	D
⋮	⋮	⋮	⋮
y	25	24	X
z	26	26	Z

That's better than multiplying by 0, but it still presents a problem when deciphering: a ciphertext B could be plaintext a or plaintext n; similarly, there are two plaintext letters for every other ciphertext letter. The same thing will happen with every other even key, so that makes 13 bad keys so far, and 13 left. There's one more bad key—take a moment to try and find it. So there are actually only 12 good keys for a multiplicative cipher, including multiplication by 1, the stupid key.

We've talked about enciphering a message with a multiplicative cipher but not really about deciphering it. Remember that to decipher a message, you need to do the opposite from enciphering it. To decrypt a Caesar cipher, you shift 3 letters left instead of shifting right. To decrypt a shift cipher, you shift k letters left. What about a multiplicative cipher? Well, you could just write out the whole table and use it backward, and in practice you probably would most of the time. But for very short messages, you might not want to write out the whole table. How can you reverse a multiplication?

The everyday answer is to divide. The opposite of multiplying by 3 is dividing by 3. That works fine for some of the letters in our multiplicative cipher with key 3. Ciphertext C becomes 3, which divided by 3 becomes 1, which is plaintext a. Ciphertext F becomes 6, which divided by 3 is 2, which is b. But what about A? It becomes 1, which divided by 3 is $\frac{1}{3}$, which isn't a letter. The solution is in the wraparound. The number 1 is the same as 27 modulo 26, so we could also say A becomes 27, which divided by 3 is 9, which is i. Likewise B could be not just 2 but also 28 and 54, and 54 divided by 3 is 18, so B corresponds to r.

ciphertext	number	divided by 3	plaintext
B	2	$\frac{2}{3}$	(not a letter)
B	28	$9\frac{1}{3}$	(not a letter)
B	54	18	r

This sort of trial and error works but is not much more efficient than writing out the table. For example, suppose your key is 15 instead of 3 for a moment. What plaintext letter does ciphertext B correspond to? Modulo 26, B could be any of the numbers 2, 28, 54, 80, 106, 132, 158, 184, 210, ...

ciphertext	number	divided by 15	plaintext
B	2	$\frac{2}{15}$	(not a letter)
B	28	$1\frac{13}{15}$	(not a letter)
B	54	$3\frac{9}{15}$	(not a letter)
B	80	$5\frac{5}{15}$	(not a letter)
B	106	$7\frac{1}{15}$	(not a letter)
B	132	$8\frac{12}{15}$	(not a letter)
B	158	$10\frac{8}{15}$	(not a letter)
B	184	$12\frac{4}{15}$	(not a letter)
B	210	14	n

It takes 9 tries before you find a value that's divisible by 15, and there's nothing to assure you that it won't be even worse for other letters. What would be really useful is a whole number that works modulo 26 like $\frac{1}{3}$ does for ordinary numbers. We could call this number $\overline{3}$. Then multiplying by $\overline{3}$ modulo 26 would be the same as multiplying by $\frac{1}{3}$ modulo 26, which is the same as dividing by 3 modulo 26.

Why might we think that $\overline{3}$ exists? If we look back at our example multiplicative cipher with key 3 from earlier, its deciphering table would look like this:

ciphertext	number	divided by 3 modulo 26	plaintext
A	1	9	i
B	2	18	r
C	3	1	a
D	4	10	j
⋮	⋮	⋮	⋮
Y	25	17	q
Z	26	26	z

It appears that perhaps dividing by 3 modulo 26 is the same as multiplying by 9 modulo 26. If this is true, then to decipher another letter, say E, we could calculate as follows:

ciphertext	number	times $\overline{3}$ = times 9	plaintext
E	5	19	s

Once I know what $\overline{3}$ is, then I can calculate this without using trial and error or searching through the encryption table.

If k is the key to a multiplicative cipher, can we be sure $\overline{k}$ exists? If so, how do we find it? Answering these questions will take us on a little detour, which, strangely enough, starts back at our "bad keys" for our multiplicative cipher.

We discovered that these bad keys are 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, and one more, which I will now reveal is 13. (You should check that this is, in fact, bad.) What these numbers have in common is that they are all multiples of 2, 13, or both. And $2 \times 13 = 26$, which is not a coincidence. If we were working with Julius Caesar's 21-letter alphabet (i.e., modulo 21), then the bad keys would be multiples of 3 or 7 (or both), since $21 = 3 \times 7$. Romanian has 28 letters and $28 = 2 \times 2 \times 7$, so the bad keys would be multiples of 2 or 7 (or both). In Danish, Norwegian, and Swedish, which have 29 letters, 29 would be the only bad key.

What we have done with these letters (26, 21, 28, 29) is to break them up into their smallest irreducible components, the **prime numbers**. This process, which is called **factoring**, can always be done in one and only one way. This was known at least as long ago as the fourth century BCE, when **Euclid** put it in his *Elements*. What we want to know is whether our key and our modulus have a **common divisor**, that is, a number that divides them both. The number 1 always divides both numbers, but that's considered trivial and doesn't count for this purpose. Euclid's *Elements* also tells us how to find a common divisor very efficiently by finding the **greatest common divisor**, or GCD, which is just what it sounds like. The method for calculating the GCD is known as the **Euclidean algorithm**, although we don't really know whether Euclid invented it or borrowed it from someone else. An **algorithm** is a well-defined method for doing something which always produces a specific correct answer for each input, such as a computer program.

Here's an example of the Euclidean algorithm in action, calculating the GCD of 756 and 210.

$$756 = 3 \times 210 + 126$$
$$210 = 1 \times 126 + 84$$
$$126 = 1 \times 84 + 42$$
$$84 = 2 \times 42 + 0$$

Each step is a division with a whole-number quotient and a remainder, just like we did earlier. The end result is that the greatest common divisor of 756 and 210 is 42, the last nonzero remainder.

We can use this algorithm to tell whether 6 is a bad key modulo 26 by calculating the GCD of 26 and 6.

$$26 = 6 \times 4 + 2$$
$$4 = 2 \times 2 + 0$$

We see that 2 is a bad key, since 2 divides 6 and 2 divides 26. What if we have a good key instead, like 3?

$$26 = 3 \times 8 + 2$$
$$3 = 2 \times 1 + 1$$
$$2 = 1 \times 2 + 0$$

The greatest whole number that divides both 3 and 26 is 1, which doesn't count, so 3 is a good key.

You might be wondering why we are bothering with Euclid's algorithm instead of just factoring the numbers and looking for prime factors in common. There are two answers to that question: First, we will eventually see that this algorithm is faster than factoring for large numbers. Second, once we have done the Euclidean algorithm, we can do a neat little trick to get $\overline{3}$.

Our next goal is to write 1 with a "3 times something" part and a "26 times something" part. We will write the equations of the Euclidean algorithm with 3 and 26 moved to the right-hand side, and every time we see part of the right-hand side without a 3 or a 26 in it, we will use a previous line to replace it with 3s and 26s.

$26 = 3 \times 8 + 2$:

$$2 = \boxed{26} - \left(\boxed{3} \times 8\right) \qquad \text{A 26 part and a 3 part, so OK.}$$

$$= \left(\boxed{26} \times 1\right) - \left(\boxed{3} \times 8\right) \qquad \text{Make both parts look the same.}$$

$3 = 2 \times 1 + 1$:

$$1 = \boxed{3} - (2 \times 1) \qquad \text{Last part has no 26, so not OK.}$$

$$= \boxed{3} - \Big(\overbrace{\left(\boxed{26} - \left(\boxed{3} \times 8\right)\right)}^{= 2 \text{ by a previous line}} \times 1\Big)$$

$$= \left(\boxed{3} \times 1\right) + \left(\boxed{3} \times 8\right) - \left(\boxed{26} \times 1\right) \qquad \text{3 parts and 26 parts.}$$

$$= \left(\boxed{3} \times 9\right) - \left(\boxed{26} \times 1\right) \qquad \text{Collect 3s and 26s.}$$

We have now written 1 with a 3 part and a 26 part. Why do we want to do this? Well, we want to work modulo 26, and 26 is the same as 0 modulo 26, so

$$1 = (3 \times 9) - (26 \times 1)$$

means that

$$1 \equiv (3 \times 9) - (0 \times 1) \quad \text{modulo } 26,$$

or

$$1 \equiv 3 \times 9 \quad \text{modulo } 26,$$

or

$$\frac{1}{3} \equiv 9 \quad \text{modulo } 26.$$

So now we have confirmed that 9 is the number $\overline{3}$, which acts like $\frac{1}{3}$ modulo 26. Again, it might seem that we could have figured this out faster by trial and error. But for large numbers, this way really is much faster.

ciphertext	number	times 9	plaintext
A	1	9	i
⋮	⋮	⋮	⋮
E	5	19	s
⋮	⋮	⋮	⋮

Incidentally, the technical term for $\overline{3}$ is the **multiplicative inverse** of 3 modulo 26. The general idea of **inverses** is terribly important in many branches of mathematics. We've now seen **additive inverses**—that is, negatives—and multiplicative inverses, and we will see other examples in the future. A good thing to notice about inverses in modular arithmetic is that, unlike in ordinary arithmetic, there isn't usually any qualitative difference between a number and it's inverse. That is, in ordinary arithmetic, 2 is a positive number and -2 is a negative number, but modulo 26, $-2 \equiv 24$. So 2 and 24 are arithmetic inverses, but neither is particularly "negative." Likewise, in ordinary arithmetic, 3 is a whole number and $\frac{1}{3}$ is a fraction, but modulo 26, 3 and 9 are multiplicative inverses, despite neither being "fractional." This is characteristic of situations where there are only finitely many numbers that are considered distinct. Another way of looking at it is that there is no real distinction between forward and backward in these situations. Likewise, there is no mathematical difference between an arbitrary encryption and an arbitrary decryption for ciphers that use these operations—once you have figured out the inverse, you can "go forward to go backward." This will be sufficiently important in later sections that you might want to think about it a bit before going on.

1.4 AFFINE CIPHERS

Now we have a shift cipher with 26 good keys, 1 of which is stupid, and a multiplicative cipher with 12 good keys, 1 of which is stupid. Both of these are pretty easy for Eve to attack with a **brute-force attack**, meaning that she just tries every possible key until she gets the right one. Even if Alice and Bob can choose either type of cipher, that still leaves Eve only 38 choices to try. But what if Alice and Bob could use more than one cipher at the same time?

This has the potential to get complicated enough so that we'll introduce a little more mathematical notation. We'll use P to stand for any number between 1 and 26 that represents a plaintext letter and C to stand for a number that represents a ciphertext letter. We'll still use k to stand for a key. Encrypting using a shift cipher with a key of k can be written as

$$C \equiv P + k \quad \text{modulo } 26,$$

and using a multiplicative cipher with a key of k can be written as

$$C \equiv kP \quad \text{modulo } 26.$$

Similarly, decrypting in the shift cipher case looks like

$$P \equiv C - k \quad \text{modulo } 26,$$

and, in the multiplicative cipher case, looks like

$$P \equiv \overline{k}C \quad \text{modulo } 26.$$

What if Alice tries to encrypt using two different shift cipher keys, say k and m?* Is that twice as secure? It would look like

$$C \equiv P + k + m \quad \text{modulo } 26.$$

Unfortunately for Alice and Bob, from Eve's point of view this looks exactly the same as encrypting once using the key $k + m$, so Eve will break the cipher just as easily if she tries a brute-force attack. The same thing will happen if Alice uses two different multiplicative cipher keys. But what if she uses one of each? Suppose Alice first multiplies the plaintext by k and then adds m to get the ciphertext:

$$C \equiv kP + m \quad \text{modulo } 26.$$

Bob will decrypt by first subtracting m and then multiplying by $\overline{k}$:

$$P \equiv \overline{k}(C - m) \quad \text{modulo } 26.$$

Notice that Bob has to not only reverse the operations, but also reverse their order! If this seems unintuitive, think about getting dressed and undressed. To get dressed, you have to put on your socks first, and then

*Cryptographers sometimes use m to stand for a second cipher key because it comes after k and the letter l looks too much like the number 1.

your shoes. To get undressed, you have to remove them both, but in the opposite order. Otherwise bad things happen.

This combination gives us a new kind of cipher, which is technically called an **affine cipher**, although I sometimes prefer to just call it a $kP + m$ cipher. There are 12 choices for k and 26 choices for m, so there are $12 \times 26 = 312$ different keys for this cipher. This is getting to be enough to make Eve's brute-force attack a little difficult, although it is still not very hard if she has access to a computer.

The idea of combining two ciphers to make a **product cipher** is a fairly obvious one and goes back quite a long time in history. The idea that one can combine any decimation method (i.e., multiplicative cipher; see Section 1.3) with any shift cipher (i.e., additive cipher, see Section 1.2) goes back at least as far as the 1930s. It's worth mentioning one much older cipher that is a particular form of a $kP + m$ cipher. This is called the **atbash** cipher, and it's at least as old as the Biblical Book of Jeremiah. Like the decimation method, it starts by writing out the plaintext alphabet. Below it, the ciphertext alphabet is the same alphabet written backward. We'll use the modern English alphabet instead of the Hebrew alphabet:

plaintext:	a	b	c	d	e	f	g	h	i	j	k	l	m
ciphertext:	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
plaintext:	n	o	p	q	r	s	t	u	v	w	x	y	z
ciphertext:	M	L	K	J	I	H	G	F	E	D	C	B	A

So why is this a form of a $kP + m$ cipher? When we translate the numbers into letters, we get

plaintext:	a	b	c	d	e	f	g	h	i	j	⋯	y	z
numbers:	1	2	3	4	5	6	7	8	9	10	⋯	25	26
some operation?:	26	25	24	23	22	21	20	19	18	17	⋯	2	1
ciphertext:	Z	Y	X	W	V	U	T	S	R	Q	⋯	B	A

We see that the ciphertext obeys the rule

$$C \equiv 27 - P \quad \text{modulo } 26.$$

Of course we can also write that as

$$C \equiv (-1)P + 27 \quad \text{modulo } 26,$$

and modulo 26 that's the same as

$$C \equiv 25P + 1 \quad \text{modulo } 26.$$

So this is a $kP + m$ cipher with the key $k = 25$, $m = 1$.

1.5 ATTACK AT DAWN: CRYPTANALYSIS OF SIMPLE SUBSTITUTION CIPHERS

If we continue along this path of making our operations modulo 26 more and more complicated, we could eventually figure out a way to specify where every single plaintext letter goes individually. So a can go to any of the 26 ciphertext letters. Then we could send b to any ciphertext letter different from the ciphertext for a, so there are 25 choices. There are 24 ciphertext letters still unused for c, then 23 for d, and so on, until we have only one letter left for z. A cipher of this kind is called a **monoalphabetic monographic substitution cipher**, **monographic** meaning that it makes substitutions one letter at a time and **monoalphabetic** meaning that the substitution rule is the same for every letter in the message. That's a pretty unwieldy name and it's a pretty common cipher, so to save time I'm just going to call it a **simple substitution cipher**. All told there are $26 \times 25 \times 24 \times \cdots \times 3 \times 2 \times 1 = 403{,}291{,}461{,}126{,}605{,}635{,}584{,}000{,}000$ ways to make this kind of cipher, which includes all three of the ciphers we have discussed as well as the cryptogram puzzles that one finds in many daily papers. That's way too many keys to attack by brute force. Unfortunately for Alice and Bob, Eve has a much better attack available to her.

A very effective way of breaking simple substitution ciphers is called **letter frequency analysis**. This technique goes back at least as far as the ninth-century Arab scholar **Abu Yusuf Yaqub ibn Ishaq al-Sabbah al-Kindi**. The idea is simply that some letters in English, Arabic, or any other human language are used more often than others. For example, in a typical English text, the letter e will occur about 13% of the time, far more than any other. If Eve has a piece of ciphertext where a

letter, say R, occurs about 13% of the time and more often than any other letter, then there's a good chance that R ($C = 18$) represents e ($P = 5$). If the cipher is an additive cipher, then Eve knows that

$$5 + k \equiv 18 \quad \text{modulo } 26,$$

so there is a very good chance that the key is $k = 13$.

If Eve has another type of cipher, such as an affine cipher, this might not be enough information. In this case, she might need to guess another letter, such as t, which occurs about 8% of the time, or a, which occurs about 7% of the time. For example, if Eve guesses that R represents e and F represents a, then she knows that

$$5k + m \equiv 18 \quad \text{modulo } 26,$$

$$1k + m \equiv 6 \quad \text{modulo } 26.$$

Now Eve has two equations in two unknowns. Subtracting them gives

$$4k \equiv 12 \quad \text{modulo } 26.$$

If the number 4 had an inverse modulo 26, then Eve could multiply each side by that inverse to cancel out the 4 and find k. Unfortunately, the GCD of 4 and 26 is 2, so 4 doesn't have an inverse. This means that our equation has either no solutions or more than one solution. If there are no solutions, it means in this case that Eve probably made a bad guess from the letter frequencies and she should try again. But in this case it turns out that there are two solutions, $k = 3$ and $k = 16$, and in either case m must be $6 - 1k$ modulo 26. So the possibilities are $k = 3$ and $m = 3$ or $k = 16$ and $m = 16$. Eve can then try to decrypt using each combination and see if she gets readable text. Since a, t, and several other letters have similar frequencies, it's possible that neither one is correct, in which case Eve has to go back to the beginning and try to guess e and a again. It might take a few guesses, but in the end Eve should be able to determine the correct key a lot faster than using brute force.

The one big caveat to this technique is that you need to have enough ciphertext to work with. The frequencies I have mentioned are only averages, and short messages may very well have radically different letter frequencies. Just imagine trying to decrypt the message "Zola is

taking zebras to the zoo," for instance. We will see how this problem can compound itself when cryptanalyzing more complicated substitution ciphers in the future.

1.6 JUST TO GET UP THAT HILL: POLYGRAPHIC SUBSTITUTION CIPHERS

There are a couple of obvious ways to make ciphers on which letter frequency analysis doesn't work—you could change the substitution rule so that it's different at different places in the message (**polyalphabetic**) or you could make the substitutions work on more than one letter at a time (**polygraphic**). Both have their places in modern cryptography, but we are going to turn now to polygraphic ciphers.

The first thing you need to decide on in a polygraphic cipher is a **block size**. Ciphers with block size 2 are digraphic, those of block size 3 are trigraphic, and so on. Digraphic ciphers were proposed as early as the sixteenth century, although the first practical ones date from the nineteenth century. In 1929, **Lester S. Hill** invented the **Hill cipher**, which can be used with any block size. We will illustrate with a block size of 2. Divide up the plaintext into 2-letter blocks. If there are unfilled spaces in the last block, fill them with any random letters—these are called **nulls**, or **padding**.

ja ck ya nd ji ll ya nd ev ex

Let the first letter in each plaintext block be P_1 and the second letter be P_2. Then calculate two ciphertext letters using the formulas

$$C_1 \equiv k_1P_1 + k_2P_2 \quad \text{modulo } 26,$$
$$C_2 \equiv k_3P_1 + k_4P_2 \quad \text{modulo } 26,$$

where k_1, k_2, k_3, and k_4 are numbers between 1 and 26, which together make up the key. For example, if the key is 3, 5, 6, 1, then the formulas are

$$C_1 \equiv 3P_1 + 5P_2 \quad \text{modulo } 26,$$
$$C_2 \equiv 6P_1 + 1P_2 \quad \text{modulo } 26.$$

If the plaintext is

plaintext:	ja	ck	ya	nd	ji	ll	ya	nd	ev	ex
numbers:	10, 1	3, 11	25, 1	14, 4	10, 9	12, 12	25, 1	14, 4	5, 22	5, 24

then the numbers for the first two letters of the ciphertext are

$$C_1 \equiv 3 \times 10 + 5 \times 1 \equiv 9 \quad \text{modulo } 26,$$
$$C_2 \equiv 6 \times 10 + 1 \times 1 \equiv 9 \quad \text{modulo } 26.$$

The x at the end of the plaintext is a null.

For the rest of the message we have

plaintext:	ja	ck	ya	nd	ji	ll	ya	nd	ev	ex
numbers:	10, 1	3, 11	25, 1	14, 4	10, 9	12, 12	25, 1	14, 4	5, 22	5, 24
Hill formulas:	9, 9	12, 3	2, 21	10, 10	23, 17	18, 6	2, 21	10, 10	21, 0	5, 2
ciphertext:	II	LC	BU	JJ	WQ	RF	BU	JJ	UZ	EB

Notice that the j of jacky gets mapped to an I, but the j of jilly gets mapped to a W. Likewise the two l's of jill get mapped to different letters, but the j and the a of jacky both end up as I's. This, of course, is because the letters are not encrypted individually, but as pairs. Also notice that yand gets mapped to BUJJ both times.

In order to decipher the message, Bob needs to solve a system of two equations in two unknowns:

$$C_1 \equiv k_1 P_1 + k_2 P_2 \quad \text{modulo } 26,$$
$$C_2 \equiv k_3 P_1 + k_4 P_2 \quad \text{modulo } 26.$$

There are lots of ways to do this; one way is to multiply the top equation by k_4 and the bottom equation by k_2 and then subtract. For instance, to decrypt the last block of our example, Bob observes that

$$5 \equiv 3P_1 + 5P_2 \quad \text{modulo } 26,$$
$$2 \equiv 6P_1 + 1P_2 \quad \text{modulo } 26,$$

which he can make into

$$1 \times 5 \equiv (1 \times 3)P_1 + (1 \times 5)P_2 \quad \text{modulo } 26,$$
$$5 \times 2 \equiv (5 \times 6)P_1 + (5 \times 1)P_2 \quad \text{modulo } 26$$

and subtract to get

$$1 \times 5 - 5 \times 2 \equiv (1 \times 3 - 5 \times 6)P_1 \quad \text{modulo } 26.$$

Similarly, Bob can multiply the top equation by k_3 and the bottom equation by k_1, which gives him

$$6 \times 5 \equiv (6 \times 3)P_1 + (6 \times 5)P_2 \quad \text{modulo } 26,$$
$$3 \times 2 \equiv (3 \times 6)P_1 + (3 \times 1)P_2 \quad \text{modulo } 26.$$

This time he takes the bottom minus the top to get

$$3 \times 2 - 6 \times 5 \equiv (3 \times 1 - 6 \times 5)P_2 \quad \text{modulo } 26.$$

Notice that in both cases there is a -27 on the right-hand side, which is $k_1k_4 - k_2k_3$. This number is called the **determinant** of the system. If the greatest common divisor of the determinant and 26 is 1, then the determinant has a multiplicative inverse, and Bob can multiply each side of his equations by that inverse to find P_1 and P_2. This is very similar to the case of ordinary arithmetic, where two equations in two unknowns can always be solved as long as the determinant of the system is not equal to zero.

In our example, the determinant is -27, as we said, which is the same as 25 modulo 26. If Bob runs through the Euclidean algorithm, he will find that

$$\overline{25} \equiv 25 \quad \text{modulo } 26,$$

so he gets

$$P_1 \equiv ((1 \times 5) - (5 \times 2)) \times 25 \quad \text{modulo } 26,$$
$$P_2 \equiv ((3 \times 2) - (6 \times 5)) \times 25 \quad \text{modulo } 26,$$

which finally reduces to

$$P_1 \equiv 5 \quad \text{modulo } 26, \qquad P_2 \equiv 24 \quad \text{modulo } 26,$$

or ex.

In general, if $k_1k_4 - k_2k_3$ has an inverse, then the solution to

$$C_1 \equiv k_1P_1 + k_2P_2 \quad \text{modulo } 26,$$
$$C_2 \equiv k_3P_1 + k_4P_2 \quad \text{modulo } 26$$

is

$$P_1 \equiv \overline{(k_1k_4 - k_2k_3)}(k_4C_1 - k_2C_2) \quad \text{modulo } 26,$$
$$P_2 \equiv \overline{(k_1k_4 - k_2k_3)}(-k_3C_1 + k_1C_2) \quad \text{modulo } 26.$$

The general form of this method for solving a system of several equations in the same number of unknowns is usually known as **Cramer's rule**, named for **Gabriel Cramer**. Cramer was an eighteenth-century Swiss mathematician who did much work studying systems of equations and the curves they describe. The same rule seems to have been published slightly earlier by **Colin Maclaurin** in Scotland. Cramer's rule is not the fastest way of solving large systems of equations, but it's certainly good enough for the block sizes one is likely to use in a Hill cipher.

Notice that if we give new names to the numbers

$$m_1 = \overline{(k_1k_4 - k_2k_3)}(k_4),$$
$$m_2 = \overline{(k_1k_4 - k_2k_3)}(-k_2),$$
$$m_3 = \overline{(k_1k_4 - k_2k_3)}(-k_3), \qquad \text{and}$$
$$m_4 = \overline{(k_1k_4 - k_2k_3)}(k_1),$$

then we can write

$$P_1 \equiv m_1C_1 + m_2C_2 \quad \text{modulo } 26,$$
$$P_2 \equiv m_3C_1 + m_4C_2 \quad \text{modulo } 26.$$

We can think of this system of equations as an inverse of the original system, and we can think of m_1, m_2, m_3, m_4 as a sort of "inverse key" for the original encryption key k_1, k_2, k_3, k_4. In our example this key would be 25×1, 25×-5, 25×-6, 25×3, or 25, 5, 6, 23 modulo 26. Once Bob has worked this out, the process of decryption works exactly the same as encryption. This is another example of the idea of going forward to go backward that we talked about in Section 1.3.

It's a little involved to work out exactly how many good keys (i.e., keys where the determinant has an inverse) there are for a Hill cipher, but it's about 45,000 for a block size of 2 and about 52,000,000,000 for a block size of 3, so a brute-force attack is getting to be rather difficult. Also note that Bob needs to be aware that there may be nulls at the end of his message. This ought to be clear when he reads it.

In 1931, Hill followed up his original cipher with several extensions. The most important one is now generally known as the **affine Hill cipher**, because it combines the original Hill cipher with an addition step, just like we combined the multiplicative and additive ciphers to get the affine cipher. If we let the block size be 2 again, the new formulas are

$$C_1 \equiv k_1P_1 + k_2P_2 + m_1 \quad \text{modulo } 26,$$
$$C_2 \equiv k_3P_1 + k_4P_2 + m_2 \quad \text{modulo } 26,$$

where the key now consists of six numbers, k_1, k_2, k_3, k_4, m_1, and m_2, all between 1 and 26. Once again, this is a good key as long as the greatest common divisor of the determinant $k_1k_4 - k_2k_3$ and 26 is 1. (The new key numbers m_1 and m_2 can be anything.) To decrypt, Bob just needs to subtract m_1 from C_1 and m_2 from C_2 and then solve the system as before.

A letter frequency analysis no longer works on a polygraphic cipher, because, as you can see from the example, the same letter in the plaintext doesn't always go to the same letter in the ciphertext. Therefore, the whole idea of guessing which letter is e fails. On the other hand, we also saw that the same plaintext block always goes to the same ciphertext block, and in the case of block size 2 or 3, it is possible to exploit this. For example, the most common digraph, or 2-letter block, is "th," which occurs, according to one study, approximately 2.5% of the time. The most common trigraph (3-letter block) is "the," which occurs, by the same study, just under 1% of the time. Eve could use facts like these to do a digraph or trigraph frequency analysis and perhaps break a digraphic or trigraphic substitution cipher. However, for larger block sizes this quickly gets very difficult, as there are a lot of possible blocks and not a lot of difference between the frequencies of the various blocks. Even in 1929, Hill managed to construct a machine that used a set of gears to mechanically encipher texts using block size 6 and was thus essentially unbreakable using frequency analysis. Unfortunately for Hill, his machine never caught on.

The Hill ciphers were never used much—they were too unwieldy to use by hand, and cryptography via mechanical devices went in the direction of polyalphabetic substitution ciphers instead. Hill's idea of using systems of equations has regained substantial importance with

the advent of digital computers in cryptography, but from a modern point of view, these ciphers used by themselves have the problem that they are badly vulnerable to a type of attack that is rather different from the ones we have talked about so far.

1.7 KNOWN-PLAINTEXT ATTACKS

So far, all of the cryptanalytic attacks we have discussed are **ciphertext-only attacks**, where all that Eve knows is the ciphertext message she has intercepted passing between Alice and Bob. But suppose that somehow Eve has gotten hold of both the plaintext and ciphertext of some message (or part of a message) that Alice has sent. Then she can try a **known-plaintext attack**, where she knows both the plaintext and the ciphertext and the goal is to get the key. Once she has the key, she can find out the content of not just the message she has, but other messages or parts of messages sent with the same key.

In the case of block size 2 and the original Hill cipher, suppose Eve recovers four letters of plaintext, P_1, P_2, P_3, and P_4, and the matching letters of ciphertext, C_1, C_2, C_3, and C_4. Then she knows

$$\begin{aligned} C_1 &\equiv k_1P_1 + k_2P_2 \quad \text{modulo } 26,\\ C_2 &\equiv k_3P_1 + k_4P_2 \quad \text{modulo } 26,\\ C_3 &\equiv k_1P_3 + k_2P_4 \quad \text{modulo } 26,\\ C_4 &\equiv k_3P_3 + k_4P_4 \quad \text{modulo } 26. \end{aligned}$$

From Eve's point of view, only the key numbers are unknowns, so she has four equations in four unknowns, and she can solve the system to recover the key.

In the earlier example, if Eve managed to recover the last two blocks of plaintext, she will know

$$\begin{aligned} 21 &\equiv k_1 5 + k_2 22 \quad \text{modulo } 26,\\ 0 &\equiv k_3 5 + k_4 22 \quad \text{modulo } 26,\\ 5 &\equiv k_1 5 + k_2 24 \quad \text{modulo } 26,\\ 2 &\equiv k_3 5 + k_4 24 \quad \text{modulo } 26. \end{aligned}$$

This is really two sets of equations,

$$21 \equiv k_1 5 + k_2 22 \quad \text{modulo } 26,$$
$$5 \equiv k_1 5 + k_2 24 \quad \text{modulo } 26$$

and

$$0 \equiv k_3 5 + k_4 22 \quad \text{modulo } 26,$$
$$2 \equiv k_3 5 + k_4 24 \quad \text{modulo } 26.$$

Eve could solve each set with Cramer's rule in the same way that Bob solved his equations in the previous section. For the first set, the rule gives

$$k_1 \equiv \overline{(5 \times 24 - 22 \times 5)}(24 \times 21 - 22 \times 5) \quad \text{modulo } 26,$$
$$k_2 \equiv \overline{(5 \times 24 - 22 \times 5)}(-5 \times 21 + 5 \times 5) \quad \text{modulo } 26.$$

If you finish the calculations, you will see

$$k_1 \equiv 3 \quad \text{modulo } 26, \qquad k_2 \equiv 5 \quad \text{modulo } 26.$$

Similarly, the second set gives Eve

$$k_3 \equiv \overline{(5 \times 24 - 22 \times 5)}(24 \times 0 - 22 \times 2) \quad \text{modulo } 26,$$
$$k_4 \equiv \overline{(5 \times 24 - 22 \times 5)}(-5 \times 0 + 5 \times 2) \quad \text{modulo } 26.$$

which gives her the last two key numbers:

$$k_3 \equiv 6 \quad \text{modulo } 26, \qquad k_4 \equiv 1 \quad \text{modulo } 26.$$

In general, Eve will need to recover only as many blocks of plaintext as there are letters in a block. So it's almost as easy to break the Hill cipher using a known-plaintext attack as it is to decipher a message. This is considered unacceptable, so the Hill cipher is never used in its original form. The idea of using a system of equations for polygraphic encryption, however, forms a piece of many modern ciphers.

1.8 LOOKING FORWARD

I warned you in the preface to this book that some of the ciphers I discuss in this book are considered obsolete in today's world, and that includes all the ciphers in this chapter and the next two, more or less. For one

thing, they all work on the letters of the alphabet, and in the modern world we want to encrypt numbers, pictures, sounds, and all sorts of other things. That's not really a serious problem, since we know how to represent all these types of information using numbers, and we can easily adjust our ciphers to use numbers instead of letters. Additive and multiplicative ciphers are vulnerable to brute-force attacks because they don't have enough keys, and with computers available to help break ciphers, affine ciphers don't really have enough keys either. Perhaps more importantly, all monoalphabetic monographic substitution ciphers are vulnerable to letter frequency attacks. Monoalphabetic substitution ciphers are significant in modern times because they are the basis for, among other things, the polyalphabetic substitution ciphers we discuss in Chapter 2. To understand that chapter, you need to understand this one first. Polyalphabetic substitution ciphers are no longer considered state of the art in security either, but we'll get to that at the end of Chapter 2.

Polygraphic substitution ciphers are resistant to frequency analysis if the block size is large enough. In fact, one of the two main types of modern cipher, the block cipher (which we define in Chapter 5), is sometimes thought of as a type of polygraphic substitution cipher acting on an alphabet of just 0 and 1. The examples we have seen so far, the Hill cipher and affine Hill cipher, are vulnerable to known-plaintext attacks, as I have shown. So these particular polygraphic ciphers are not considered secure. As I mentioned, however, these two ciphers are used as building blocks in modern block ciphers, including the current US government standard in block ciphers, which I describe in Chapter 4. So you can't understand modern block ciphers without the affine Hill cipher, and you can't properly understand that without the additive, multiplicative, and affine ciphers.

I should point out that the cryptanalysis techniques discussed in this chapter, while not state of the art, are still very important in understanding modern cryptanalytic techniques. Letter frequency is not relevant with regard to a modern block cipher, but frequency attacks certainly are. The differential attacks discussed in Chapter 4, for instance, rely heavily on the same sorts of statistical frequency calculations as letter frequency attacks, but applied to the differences between ciphertexts rather than the ciphertexts themselves. Similarly, the linear attacks that

I mention in Chapter 4 are more sophisticated versions of the known-plaintext attack I showed you against the Hill cipher. Modern ciphers do not consist solely of the types of equations that the Hill and affine Hill ciphers do, but they can sometimes be approximated by such equations. Linear cryptanalysis takes advantage of this fact.

Finally, you might be wondering whether the concepts and notation of modular arithmetic were really necessary, or whether there were easier ways to describe the ciphers in this chapter. Additive, multiplicative, and affine ciphers were in fact used and analyzed perfectly well before anyone thought to describe them with modular arithmetic. The Hill and affine Hill ciphers, on the other hand, were invented with modular arithmetic in mind and are harder to deal with without those concepts. Even more importantly, modular arithmetic is critical for understanding the exponential ciphers and public-key ciphers of Chapters 6, 7, and 8.