V'ir Tbg n Frperg

An Introduction to Cryptography

Scott Sutherland*

October 2005

Abstract

We introduce ideas and techniques from cryptography, using mathematics related to the middle-school mathematics curriculum. The majority of the material in this note was presented as part of a professional development workshop for middle-school mathematics teachers.

1 Pleased to Meet You, Hope You Guessed My Name

People have been interested in writing secret messages almost as long as people have been able to write. During the twentieth century, it became clear that cryptography had a lot to do with mathematics, and the role of making and breaking codes has increased dramatically in the military since the First World War.[1] Although you may not know it, cryptography plays a role in most of our daily lives as well. When we make a purchase over the internet, our web browser uses encryption to keep our financial data secure. Automatic Teller Machines (ATMs), satellite TV, and a number of other everyday services use encryption. Secret codes are also just kind of fun to fiddle with, and fun is part of our main focus today.

First, let's introduce some terminology. The discipline of encoding and decoding secret messages is called **cryptography**, from the Greek words *kryptos* ($\kappa\rho\upsilon\pi\tau o\sigma$), meaning hidden or secret, and *graphia* ($\gamma\rho\alpha\phi\iota\alpha$), meaning writing. The original, readable message

*Institute for Mathematical Sciences, Stony Brook University, Stony Brook, New York, 11794-3660. `scott@math.sunysb.edu`, `http://www.math.sunysb.edu/~scott/`

[1] Many people have claimed that if the Allies hadn't broken the German Enigma code, the outcome of the Second World War would have been different.

is referred to as the **plaintext** or **clear text**. Encoding the message is called **encryption** (or sometimes **enciphering**), and the hard-to-read result is called the **ciphertext** (or the **encrypted message**). Turning the ciphertext back into plaintext is called **decrypting**, **deciphering**, or **decoding** the message. Usually the conversion process depends on an additional piece of information, usually called the **key** or the **password**; the key forms the basis of the security and must be kept private. Sometimes there are different keys used to encrypt and decrypt messages. If it is infeasible to determine the key used for decryption without knowing the key used to encrypt, this method can be used for **public-key cryptography** (which we won't be able to discuss). Finally, decrypting a message without being told the corresponding key is called **cracking** or **breaking** the code.

Technically speaking, there is a difference between a cipher and a code: a **code** works at the level of meaning, replacing words or phrases with others (for example, when parents refer to a four-year-old's upcoming birthday party as "the festivities" to avoid exciting her prematurely), while a **cipher** replaces individual letters or groups of letters with others. Naturally, these are not mutually exclusive, and are often used together.[2] While formally they are different things, one often informally uses the word "code" when speaking of ciphers. Since we will only be looking at ciphers, we will use the terms interchangeably.

One last bit of terminology we must cover is the choice of **Alphabet** for the plaintext (and the ciphertext). We need to agree up front which characters we will be writing our messages in. Prior to the advent of computers, messages typically consisted of only the 26 letters of the alphabet; all spaces, punctuation, numerals, and so on were removed from the message prior to encryption. No distinction was made between upper-case and lower-case letters. Today, a computer tends to do the enciphering and deciphering, and the alphabet is much larger, typically containing 128, 256, or more character codes. We'll adopt the more spy-like convention of using just 26 characters.

In this note, we'll write our plaintext in all lower case letters, as

```
abcdefghijklmnopqrstuvwxyz
```

and our ciphertext in all uppercase letters

[2] Another way to create a secret message is using **steganography**, which is the process of hiding the fact that a message exists at all. Using "invisible ink" is steganography, as is hiding messages within other messages. For example, looking at the second letter of each word in the message below (which was reportedly sent by a German spy during World War I [Kah, p. 521]),

```
   Apparently neutral's protest is thoroughly discounted and ignored.
Isman hard hit.  Blockade issue affects pretext for embargo on
byproducts, ejecting suets and vegetable oils.
```

one finds a rather different message, namely

```
Pershing sails from NY June I.
```

Steganography is often used to augment cryptography, and is also related to "digital watermarking".

`ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Since we'll be leaving out the spacing and punctuation, it is traditional to group the text into blocks of letters (typically five), so that one doesn't get lost.

For example, if we wanted to encrypt the sentence "Please do not read this.", we would first represent it as

`pleas edono tread this`

Note that the last group has only four letters, because the length of the message was not divisible by 5.

2 I'm a Substitute for Another Guy

The simplest ciphers are based on the idea of replacing each letter in the plaintext with a different letter. For example, instead of **`a`** we might use **`R`**, for **`b`** write **`A`**, and so on. This class of ciphers is called a **substitution cipher** (more precisely, a monoalphabetic substitution cipher); these are the types of ciphers you sometimes see on the crossword page of the newspaper. Note that in this case, the key is rather long (namely, just as long as the alphabet, since we must say which character gets replaced with which), and there are many possible encryptions (26! or $40,329,146,112,660,563,584,000,000$). But, they are still pretty easy to crack.

Let's look at an example.

`RUQDY QNTRZ AQIQB NAZNC YQQBQ WBJQR UJWXS RUNVW WENCQ TRYQK QRK`

At first glance, this may seem like total gibberish. But upon a little examination, we see that the letter **`Q`** occurs 11 times, much more than any other. It would reasonable to guess that **`Q`** stands for a commonly occurring letter. In English, the letter **`e`** is the most common letter, followed by **`t`**, **`a`**, and **`o`**.[3]

So, in our message, we would likely guess that **`Q`** corresponds to **`e`**, and that **`W`**, **`R`**, and **`N`** correspond to **`o`**, **`a`**, and **`t`**, although these last three might be permuted. Then we plug them in, and guess at which letters correspond to which until the message makes sense. Once we have a few letters, we can try to recognize patterns corresponding to common words (**`the`**, **`and`**, and so on). We can also make use of the fact that the most common pairs of letters in English are (in order) **`th`**, **`he`**, **`in`**, **`en`**, **`nt`**, **`re`**, **`er`**, **`an`**, **`ti`**, **`es`**, **`on`**, **`at`**, **`se`**, **`nd`**, **`or`**, **`ar`**, **`al`**, **`te`**, **`co`**, **`de`**, **`to`**, **`ra`**, **`et`**, **`ed`**, **`it`**, **`sa`**, **`em`**, and **`ro`**.

Making use of such probabilistic guesses is called **frequency analysis**. As the length of the ciphertext increases, frequency analysis becomes increasingly reliable. There are

[3] This frequency sometimes shows up as **`etaoin shrdlu`** in pre-1980s print, because the keys on Linotype machines used to set type were arranged in frequency order. For the same reason, **`qwerty`** or **`asdf`** show up when people bang on keyboards.

Letter	Frequency	Letter	Frequency	Letter	Frequency
a	8.167	j	0.153	s	6.327
b	1.492	k	0.772	t	9.056
c	2.782	l	4.025	u	2.758
d	4.253	m	2.406	v	0.978
e	12.702	n	6.749	w	2.360
f	2.228	o	7.507	x	0.150
g	2.015	p	1.929	y	1.974
h	6.094	q	0.095	z	0.074
i	6.966	r	5.987		

Table 1: Frequency (in percent) of occurrence of letters in English text (adapted from [Lew])

records from the 9th century A.D. indicating that frequency analysis was in use in the Arab world at that time. For this method to work, typically you need a message of at least 50 characters.

So far, we have what is below (assuming the guesses for characters are correct):

```
RUQDY QNTRZ AQIQB NAZNC YQQBQ WBJQR UJWXS RUNVW WENCQ TRYQK QRK
a e   et a   e e  t  t   ee e o  ea   o   a t o o t e  a e  ea
```

It still takes quite a bit of playing around to get the rest of the message. Remember that the spaces were removed from the original; as a hint, I'll put them back in, together with our guesses. To avoid spoiling the fun for those who don't want a hint, it is in a footnote.[4]

3 The Caesar Cipher and Modular Arithmetic

In order to better understand substitution ciphers, and make something better out of them, we do what is very common in mathematics: we make the problem easier, and don't worry (at first) that it seems like a step backwards.

So, instead of considering all possible substitutions, we just think about some simple ones: those where we leave the alphabet in order, and just shift by a few letters. Julius Caesar is reported to have used such a cipher with a shift of 3 for his military communications.

[4] A hint:

```
R UQDYQN TRZ AQ IQBN AZ NCYQQ BQWBJQ RU JWXS RU NVW WE NCQT RYQ KQRK
a  e  et  a   e  e t    t  ee  eo  e a   o   a  t o o  t e  a e  ea
```

The correspondence between plaintext and ciphertext is as follows:

```
abcde fghij klmno pqrst uvwxy z
XYZAB CDEFG HIJKL MNOPQ RSTUV W
```

If we encrypt `veni vidi vici` using the Caesar cipher, we get `SBKF SFGF SFZF`. To decrypt, we just shift back by 3 letters.

So where's the math? Let's see if we can find it.

First, we assign a number to each letter of the plaintext alphabet, beginning with 0, so we that have the correspondence below.

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
0	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

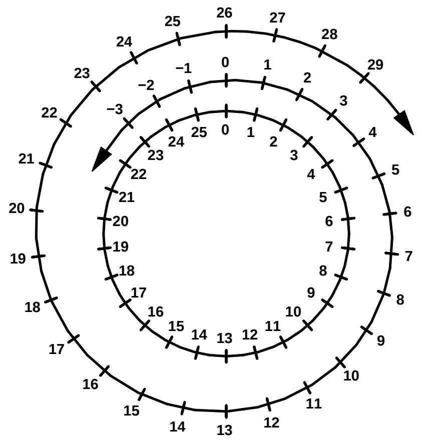
Now, we can view our enciphering as taking the number corresponding to the letter, adding 3 to it, and then writing down the letter that corresponds to the sum. If the result is 26 or larger, we subtract 26 so that if falls back in the desired range.

This type of arithmetic is called **addition modulo 26** or just adding **mod 26**. The number 26 is called **the base**. You are already quite familiar with another type of modular arithmetic, namely working mod 12. When we answer a question like "What time will it be 4 hours after 10 o'clock?", we are adding mod 12. Another way to view modular arithmetic is that we do "regular" arithmetic, divide the answer by the base, and then only keep the remainder.[5] We sometimes also work modulo 7 when we make statements like "My birthday is on a Friday this year, so next year it will be on Saturday"– here we are using the fact that $365 = 7 \times 52 + 1$, so $365 \equiv 1 (\text{mod } 7)$. Waiting a year advances the day of the week by one (except in a leap year).

Returning to our encryption problem, we see that applying the Caesar cipher corresponds just to adding 3 (mod 26). To decipher, we can either subtract 3 modulo 26 (remembering to add 26 to the answer if it turns out negative), or we can add 23, which is $26 - 3$. These give exactly the same answer after reducing modulo 26, in the same way that a clock will read the same if you move the hands ahead by 3 hours or back 9 hours.

[5] Modular arithmetic works for multiplication as well as for addition and subtraction. One has to be careful about what division means, however. We'll cover this further in section 6.

A useful mental model for modular addition is a "number circle". Take the familiar number line and wrap it around a circle so that the base (26) falls on zero. Then adding corresponds to a clockwise rotation, and subtraction to counterclockwise rotation.



Naturally, we don't have to shift by 3 as Julius Caesar did (apparently Augustus Caesar preferred to shift by 1). We have 25 possible ciphers like the Caesar cipher, which are called **shift ciphers**, or sometimes the general shift cipher is called "a Caesar cipher".[6]If someone refers to *the* Caesar cipher, they almost certainly mean a shift cipher with a shift of 3.

Notice that shift ciphers are *very* easy to break, since you only have to guess one letter and then you know everything. If you haven't already discovered it, the title of this note is encrypted with a shift cipher, leaving the spaces and punctuation in. So that you don't have to look back, the (encrypted) title is

`V'IR TBG N FRPERG`

4 Many Caesars: the Vigenère Cipher

So far, it seems we've gone in the wrong direction: from the poor security offered by the general substitution cipher to nearly no security offered by a shift cipher. But, taking a step backward will allow us to leap forward.

In the Vigenère Cipher, we choose a word or phrase as our encryption key. We then convert it to a sequence of numbers (again using **`a`**=0, **`b`**=1, . . . , **`z`**=25), and apply a different shift to the plaintext corresponding to the letters in the keyword. When we use up the shifts from the keyword, we repeat it again.

This is probably best understood via an example. Suppose we choose **`jasmine`** as our key, and want to encrypt the phrase **`we are getting hungry`**. Translating **`jasmine`** to its numerical equivalents gives the sequence of shifts 9, 0, 18, 12, 8, 13, 4. So we work as follows:

[6] One shift cipher commonly used in internet postings or emails that may tell the ending of a movie or the answer to a puzzle is called "rot13" (for rotation by 13). This is just a Caesar-like cipher with a shift of 13 (leaving all the non-alphabetic text alone). This is popular because applying the same shift again decodes the text.

	w	**e**	**a**	**r**	**e**	**g**	**e**	**t**	**t**	**i**	**n**	**g**	**h**	**u**	**n**	**g**	**r**	**y**
	22	4	0	17	4	6	4	19	19	8	13	6	7	20	13	6	17	24
$+$	9	0	18	12	8	13	4	9	0	18	12	8	13	8	4	9	0	18
$=$	5	4	18	3	12	19	8	2	19	0	25	14	20	2	17	15	17	16
	F	**E**	**A**	**D**	**M**	**T**	**I**	**C**	**T**	**A**	**Z**	**O**	**U**	**C**	**R**	**P**	**R**	**Q**

Depending on where it falls in the message, each character of plaintext could be any one of several ciphertexts. For example, in the example above, **e** becomes **E**, **M**, or **I**, and the two **E**s in the ciphertext come from an **e** or an **r**.

The Vigenère cipher is significantly harder to break than a Caesar cipher. For about 300 years, it was believed to be unbreakable, although Charles Babbage and Friedrich Kasiski independently determined a method of breaking it in the middle of the nineteenth century. The method uses repeated patterns in the text to determine the length of the key. Once it is known that the key is, say, 8 characters long, frequency analysis can be applied to every 8th letter to determine the plaintext. This method is called **Kasiski examination** (although it was first discovered by Babbage). Despite the fact that the method of breaking the Vigenére cipher had been published 50 years earlier, the cipher was widely held to be unbreakable until the 1920s, being described as "impossible of translation" in an article appearing in *Scientific American* in 1917.

We should remark that one can implement this cipher using a *tabula recta*, which is a 26×26 table listing all the shifts corresponding to each letter of the key. It is really just an addition table for letters. To use the *tabula recta*, the plaintext letter gives the row, and the key letter gives the column. The letter of ciphertext is then found where the two intersect. Using such a table, one doesn't need to do the numerical conversion and the modular addition; this is how people in the sixteenth century used the cipher. However, the underlying mathematical structure is hidden, which is a disadvantage for our purposes.

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

Table 2: A *tabula recta*

The Vigenère cipher is named after Blaise de Vigenère, a sixteenth century diplomat,although it was in fact invented by Giovan Batista Belaso in 1553. Vigenère actually invented a different cipher, which we will now discuss.

The cipher Vigenère invented is a little harder to crack than the one that bears his name. In this cipher, rather than repeating the key, one begins with a key, and then appends the text itself to get the shifts to apply. This is a type of **autokey** cipher, because the key is (semi-)automatically generated by the message itself.

As an example, we will use the same plaintext (`we are getting hungry`) and initial key (`jasmine`), and apply Vigenère's autokey cipher. The autokey will then be `jasminewearegettin`, giving us the encryption below.[7]

	w	e	a	r	e	g	e	t	t	i	n	g	h	u	n	g	r	y
	22	4	0	17	4	6	4	19	19	8	13	6	7	20	13	6	17	24
+	9	0	18	12	8	13	4	22	4	0	17	4	6	4	19	19	8	13
=	5	4	18	3	12	19	8	15	23	8	4	10	13	24	6	25	25	11
	F	E	A	D	M	T	I	P	X	I	E	K	N	Y	G	Z	Z	L

Another variation on these same ideas is to use a passage from a book in order to determine the sequence of shifts. The location of the start of the passage serves as the key. For example, the key might be given as 123:5, and then cryptographer turns to page 123 and begins with the 5th word to extract the key. The Bible and the works of Shakespeare were both popular to use as source books.

While these methods do away with the problem of repeating shifts that hinder the Vigenère cipher, frequency analysis and similar techniques can still be employed. This is because certain letters (specifically `e`, `t`, `o`, etc.) will still occur more frequently in both the text and the key. Given a long sample of ciphertext, a competent cryptographer can break both of these codes.

5 An Unbreakable Cipher

Is it possible to create a cipher that no one can break, no matter how smart they are and how hard they try? Yes, as long as you remember that you don't get to know the key to crack the code.

The idea is similar to the Vigenère cipher, except that instead of using English text as our key, we use a sequence of random numbers. It is critical that the random sequence of numbers be truly random, and that the sequence never be reused. Such a cipher is called a **one-time pad**, and was proven to be unbreakable by Claude Shannon in 1949.

[7] Some accounts append the ciphertext to the initial key rather than the plaintext as we do here. In both cases, there is a serious drawback to this cipher: if an error is made in enciphering even one letter, it will propagate throughout the remainder of the text.

To use a one-time pad, we have an arbitrarily long list of random numbers. This sequence is our "one-time pad". We add each of these in turn to our plaintext, and reduce modulo 26. This gives our ciphertext. To decrypt, we reverse the process, subtracting the same sequence of numbers from the ciphertext. For this to be truly secure, we can never use this sequence of numbers again, hence the phrase "one-time" in the name.

The security of the system stems from the fact that any plaintext can encrypt to any ciphertext of the same length. For example, the ciphertext **`QQQQQQ`** could correspond to the plaintext **`attack`** or **`gohome`**. For the first, the key sequence is 16,23,23,16,14,6 and in the second message, the key is 13,2,12,2,4,9. Since the key could be any of these (or any other one), there is no way to break the cipher except to get your hands on the key.

One-time pads were used heavily during the second world war, and during the cold war. Books consisting of long lists of random digits were given to agents. But it was critical that the codebooks not fall into enemy hands, or the ciphers would immediately become useless. Anyone who possessed the codebook could easily decrypt the messages.

It is also very important that the numbers be truly random. A computer program cannot generate truly random numbers without some outside source of randomness. Several successful attacks on early "secure" web communications relied on weaknesses in the browser's random number generator. There is a story that during World War II, some of the codebooks were generated by drawing numbered balls out of a bin. However, the person drawing the balls would put the ball back if he drew it twice in a row, thinking a repeated number wasn't random. This slight deviation from true randomness was enough to enable several "unbreakable" messages to be broken.

6 *Mod*ern Times [8]

Having squeezed all of the juice out of the idea of adding letters to others, we go back and look again at other ways to proceed. It seems like it should work to multiply the numeric codes for the letters in the plaintext instead of adding them, so let's try that. Just as before, we'll begin with something analogous to the Caesar cipher: our key is a single number between 1 and 25, and we multiply each letter by that number and reduce modulo 26.

Since the Caesar cipher added 3, let's try multiplying by 3. Suppose our plaintext is **`blue`**. Representing this as numbers, we have the sequence 1,11,20,4 (refer to the chart on page 5 if you like). Now we multiply each one by 3 and reduce modulo 26. For the first letter of the ciphertext, we get the numeric value 3, or **`D`**. The second letter is 33 mod 26, or 7, which we can write as **`H`**. The third is 60 mod 26, which is 8 (or **`I`**), since $60 = 8 + 2 \times 26$. And our final letter encrypts to **`M`**, so the ciphertext is **`DHIM`**.

[8] Because of time limitations, this and following sections were not covered in the professional development workshop.

One thing to notice here is that we get a different class of substitution ciphers out of this process. The Caesar ciphers gave us one set, and these multiplicative ones give us another set. These seem to mix things up a little more (for example, although **l** and **u** start out far apart in the alphabet, they encrypt to the adjacent letters **H** and **I**.

Now that we see how to encrypt, let's try to decrypt this enciphered message, assuming we know that the encryption key was 3. That is, we are given the ciphertext **DHIM** as well as the fact that the encryption key was 3, and we want to recover the plaintext. Writing the ciphertext as numbers, we have 3,7,8,12 and we want to divide each by 3. The first is easy: $3/3 = 1$, and sure enough, **D** corresponds to **b**. But $7/3 = 2\frac{1}{3}$, and we don't have a letter for that. Is all lost?

We just have to make sense of what we mean by division. Remember, we are working modulo 26, so we want to solve the equation

$$7 = 3x (\text{mod } 26).$$

This means we get to add multiples of 26 to get things to work out; the above equation is the same as saying

$$7 = 3x \quad \text{or} \quad 7 + 26 = 3x \quad \text{or} \quad 7 + 52 = 3x \quad \text{or} \quad \ldots$$

Since $7 + 26 = 33$, we see that $x = 11$ works, and our plaintext letter was indeed the letter **l**.

Deciphering **I** works the same way: $8/3$ isn't a whole number, and neither is $34/3$, but $60/3 = 20$, so our plaintext is just the letter with character code 20, or **u**.

We could have saved a little work by changing division into an appropriate multiplication. Just as with real numbers, to divide by 3 we can multiply by $\frac{1}{3}$, as long as we interpret $\frac{1}{3}$ correctly. As above, we have to find a number x so that

$$3x = 1(\text{mod } 26).$$

This number is 9, since $3 \times 9 = 27 = 1 + 26$. 9 is called the **multiplicative inverse** of 1, modulo 26. When we are working modulo 26, division by 3 is the same as multiplication by 9.

Now suppose we had tried a key of 4, and wanted to encrypt the name of our friend **bob**. This has the numeric sequence 1,14,1, and after multiplying by 4, we have 4,56,4. Since we are working mod 26, we must reduce 56 to 4, giving the ciphertext **EEE**. Something is fishy here— there is no way to know whether the **E** came from **b** or **o**. Why is 4 different from 3?

The issue is 4 and 26 share a common divisor, 2, while 3 and 26 are relatively prime. When the key (4) and the size of the alphabet (26) are not relatively prime, there will be different plaintext letters which correspond to the same ciphertext. In the example above, we have

$$4 \times 1 = 4 \qquad \text{and} \qquad 4 \times 14 = 56 = 4 + (2 \times 26).$$

If any time we use key which is an even number, any two plaintext letters which have character codes differing by 13 will encrypt to the same ciphertext, and so can't be deciphered again.

Notice also that 4 does not have a multiplicative inverse mod 26, because the equation $4x = 1(\text{mod } 26)$ has no solution. To see this, notice that $4x$ must be an even number, but if we add 1 to any multiple of 26, we get an odd number. So there can be no number x so that $4x$ is of the form $1 + 26k$.

There are two obvious ways to avoid this collision problem: don't use keys which share a common divisor with the length of the alphabet, or change the the alphabet so that its length is a prime number. In our case, this is easily accomplished by adding some punctuation to the alphabet (for example, adding a period, a comma, and and exclamation mark), which would give us a 29-character alphabet. Or we could include the digits 0–9, and a period. In order to be consistent with the what we've done before, however, we'll just avoid using even keys (or 13).

We can combine the cipher above with a Caesar cipher to get an **affine cipher**. That is, our key consists of two numbers m and b, where m is relatively prime to the length of the alphabet. Then to get the ciphertext for the letter with character code x, we compute $mx + b$ and reduce modulo the length of the alphabet (26 in our case).

As an example of an affine cipher, let's encrypt the work **`blue`** using $m = 3$ and $b = 20$. As before, we note that **`blue`** has character codes 1,11,20,4, which becomes 3,7,8,12 after multiplying by 3 (and reducing/ mod 26). Now we add 20 to each one to get 23,27,28,32, which reduces to 23,1,2,6 modulo 26. This gives the ciphertext **`XBCG`**.

To decrypt, we just reverse the process: subtract 20 from each, then multiply by 9 (the inverse of 3), reducing modulo 26.

An affine cipher is just a specific case of the general substitution cipher we discussed in section 2, and so it is readily broken using frequency analysis. Indeed, if you know a message is encrypted with an affine cipher, you only need to determine what two letters are to crack the encryption.

7 Twins and Triplets

Of course, one could increase the security of an an affine cipher using a different one on each letter, as the Vigenère cipher discussed in section 4 does. But this is extra effort with no payoff: such a cipher can be cracked by exactly the same methods, and it is more work to encipher.

However, because an affine cipher sends adjacent letters far apart, one way to improve the cipher is to work on multiple letters at a time, either in pairs, triplets, or larger groupings. Instead of assigning numbers to individual letters, we think of our message as made up of "super-characters", where each individual letter gives us one "digit" of the group.

For example, if we wanted our super-characters to be made up of pairs of two letters (called a **digraph**), we would think of the plaintext **`double`** as being made up of the three pairs of digraphs **`do`**, **`ub`**, and **`le`**. How many pairs of such digraphs do we have? Since we have 26 choices for the first letter and 26 for the second, there are $26 \times 26 = 676$ such pairs. The first one is **`aa`** and gets assigned the number 0, **`ab`** gets 1, **`ac`** has the code 2, and so on, up to **`zz`** which gets the character code 675. If the plaintext has an odd number of characters, we add an additional character onto the end (usually **`x`**, **`q`**, or **`z`**).

With so many digraphs, we certainly don't want to have to use a table to look up the numeric equivalences. But notice that we can readily figure out which letter pair gets which number by paying attention to place value. Remember that when we write the decimal number for six hundred seventy five, we write

$$675 = 6 \times 100 + 7 \times 10 + 5.$$

Here, we don't have ten "digits", we have twenty-six.[9] So to determine the code for **`do`**, we multiply the number for the single letter **`d`** by 26, and add the number for **`o`**. That is, it has the numeric code $3 \times 26 + 14 = 92$. Similarly, **`ub`** corresponds to $20 \times 26 + 1 = 521$, and **`le`** gives 290. To go from the number to its letter equivalent, we divide by 26; the dividend gives the first character, and the remainder gives the second. For example, to figure out the letters which correspond to 524, we divide 524 by 26. The result is 20, with a remainder of 4, so 524 corresponds to **`ue`**.

Let's encrypt the word **`blue`** using the same affine cipher as in section 6 (that is, with $m = 3$ and $b = 20$), but working with digraphs.

plaintext	**`bl`**	**`ue`**
x	$1 \times 26 + 11 = 37$	$20 \times 26 + 4 = 524$
3x+20	131 $131 = 5 \times 26 + 1$	$1592 = 240 \pmod{676}$ $240 = 9 \times 26 + 6$
ciphertext	**`FB`**	**`JG`**

[9] I suppose we could call these hexicosimal numbers (from the Greek for 26), but that's too weird, so we just say we are working in base 26.

Note that we can take m to be any number between 1 and 675 (as long as it is relatively prime to 26), and we can take b as anything between 0 and 675.

To break such a code, we can apply frequency analysis, but this time we must look at frequency of pairs of letters. This is still doable, but more challenging.

Notice that we can just as well use triplets of letters (**trigraphs**), or blocks of any length we choose. For example, if we were using 4-tuples, the word **`blue`** would be a single "character" and would have the numeric code

$$1 \times 26^3 \ + \ 11 \times 26^2 \ + \ 20 \times 26 \ + \ 4 \quad = \ 25536.$$

However, such numbers quickly become unwieldy unless one is using a computer.

8 The Hill Cipher

Rather than working with such large numbers, the **Hill cipher** works on groups of letters in a somewhat different manner. The Hill cipher works by viewing a group of letters as a vector, and encryption is done by matrix multiplication. However, we will avoid reference to linear algebra in our explanation.

The Hill cipher was first described by Lester Hill in a paper published in 1931. While this cipher can work on blocks of letters of any length, we'll describe it as working on pairs of letters, or digraphs.

First, our key consists of four numbers which we call a, b, c, and d. These numbers must be chosen so that the quantity $ad - bc$ is relatively prime to the length of the alphabet (26 in our case, so $ad - bc$ cannot be even or a multiple of 13).

To encrypt a pair of letters, we look up their numeric equivalents as usual. Suppose these numbers are x and y. Then the corresponding letters in the ciphertext are given by

$$(ax + by) \bmod 26 \quad \text{and} \quad (cx + dy) \bmod 26.$$

As an example, let's encipher the phrase **`do it now`** using a Hill Cipher with the key $a = 2$, $b = 3$, $c = 5$, and $d = 6$. Since $ad - bc = -3 = 23 (\text{mod } 26)$, this is a valid key. To encode, we break our text up into pairs, and since there are an odd number of letters, we add **`x`** to the end.

plaintext	**`do`**	**`it`**	**`no`**	**`wx`**
`(ax+by, cx+dy)`	$(3,14)$ $(6+42=22$ $15+84=21)$	$(8,19)$ $(16+57=21,$ $40+114=24)$	$(13,14)$ $(26+42=16,$ $65+84=19)$	$(22,23)$ $(44+69=9,$ $110+138=14$
ciphertext	**`WV`**	**`VY`**	**`QT`**	**`JO`**

To decipher, we apply a different Hill cipher to the ciphertext. If the pairs of letters have numeric equivalencies X and Y, then the plaintext is given by

$$(AX + BY) \bmod 26 \quad \text{and} \quad (CX + DY) \bmod 26.$$

To find A, B, C, and D, we first find the multiplicative inverse of $ad - bc$ and denote it by J. Then[10]

$$A = (d \times J) \bmod 26, \qquad B = (-b \times J) \bmod 26$$
$$C = (-c \times J) \bmod 26, \qquad D = (a \times J) \bmod 26$$

For example, if we encipher with $a = 2$, $b = 3$, $c = 5$, and $d = 6$ as above, then $J = 17$, and the deciphering transformation has $A = 24$, $B = 1$, $C = 19$, and $D = 8$.

The real strength of the Hill cipher is that it can be adapted to work on large blocks of text without having to use huge numbers as we did in the previous section. For example, to work on blocks of 3 letters at a time, we choose 9 numbers $a, b, c, d, e, f, g, h, k$, and then the code of the ciphertext corresponding to the triple (x, y, z) is

$$(ax + by + cz, dx + ey + fz, gx + hy + kz) \bmod 26.$$

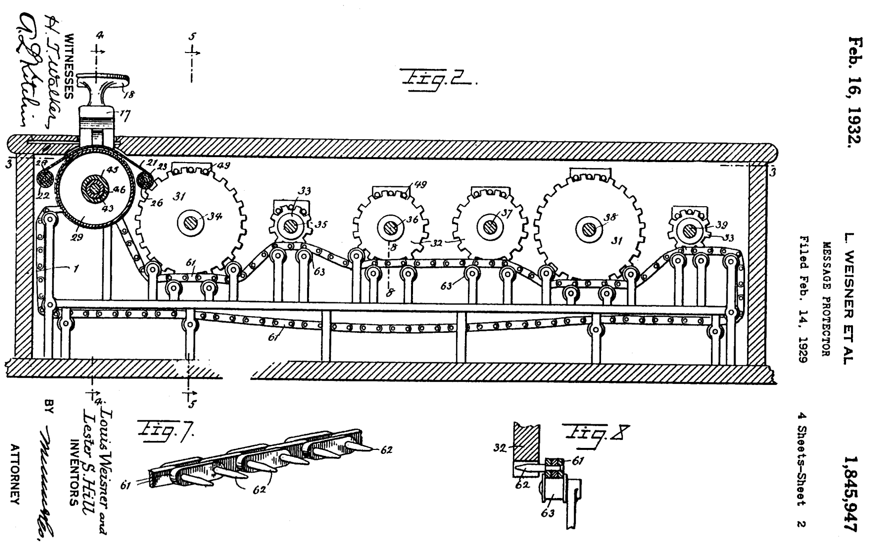
Extending this to larger blocks of letters works analogously.

In attempting to crack a Hill cipher, the usefulness of frequency analysis becomes vanishingly small as the size of the blocks of letters increases. A Hill cipher isn't hard to implement on groups of 6 or more letters. A major disadvantage, however, is that it is quite susceptible to what is called a **known plaintext attack**. If part of the plaintext is known to the attacker, then the deciphering transformation can be deduced merely by solving some linear equations. Some help can be gained by applying a Vigenère cipher after encrypting with a Hill cipher,[11] although there are much better ways to improve the security.

It is possible to build a machine that does all of the arithmetic corresponding to the Hill cipher by means of gears and pulleys. Lester Hill and Louis Weisner were issued a patent for a "Message Protector" which mechanically implemented a Hill cipher acting on blocks of six letters at a time. The major gears in the machine correspond to a "number circle" as discussed on page 6; multiplication is done by advancing the gear by a fixed number of teeth several times (that is, repeated addition). The figure is taken from the patent application (adapted from [Cdb]).

[10] This formula comes from linear algebra. It is just the inverse of the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

[11] If the length of the key for the Vigenère cipher is the same as the length of the letter-blocks, this yields an affine cipher $AX + B$, where A is an $n \times n$ matrix, X is our vector of n plaintext letters, and B is the n-vector corresponding to the Vigenère key.



9 Adieu

In this note we've discussed just a few ideas related to cryptography. There are many others, most of which use quite sophisticated mathematics. The National Security Agency, which is the U.S. government agency dealing with cryptography, is reportedly also the largest employer of mathematicians in the world. (We don't really know, because they keep this information secret!)

References

[Cdb] *Crypto Drop Box*. `http://www.und.nodak.edu/org/crypto/crypto/`.
Contains figures from Hill's patent application, pictures of a German Enigma machine, and other material. Maintained by the American Cryptogram Association.

[Fla] Sarah Flannery with David Flannery, *In Code: A Mathematical Journey*. Workman Publishing, 2001.
A very engaging and readable account by Sarah Flannery, whose high school project on cryptography (done when she was 16) won the 1999 European Union Young Scientists contest. Part memoir, part puzzle book, and part mathematical exploration, and all very entertaining.

[Gar] Martin Gardner, *Codes, Ciphers and Secret Writing*. Dover Publishing, 1984.
Explains in simple terms how to encode and decode messages in various ciphers, de-

scribes coding machines, and gives formulas for invisible ink. Appropriate for middle school students.

[Kah] David Kahn, *The Codebreakers; The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Charles Scribner's Sons, 1996.
A massive (nearly 1200 pages!), but very readable historical account of cryptography. A classic.

[Lew] Robert E. Lewand, *Cryptological Mathematics*. Mathematical Association of America, 2000.
Covers the essential number theory required to understand various encryption schemes. Requires mathematics at about the level of a good high-school student.

[Sut] Scott Sutherland, *fsqFsHn sGGousG (Secret Messages)*.
`http://www.math.sunysb.edu/~scott/Book331/crypto.html`.
Part a text aimed at doing mathematics with computers, for math majors in their junior or sophomore year of college. Uses the `Maple` *programming language.*