$$(x+y)^3 = (x+y)\cdot(x+y)\cdot(x+y), \tag{4.2}$$

the result is a sum of terms x^3, x^2y, xy^2, and y^3. There is only one x^3 term, since to get x^3 we must take x from each of the three factors in (4.2). How many copies of x^2y are there? We can get x^2y in several ways. For example, we could take x from the first two factors and y from the last factor. Or we could take x from the first and third factors and take y from the second factor. Thus we get x^2y by choosing two of the three factors in (4.2) to give x (note that the order doesn't matter), and then the remaining factor gives y. There are thus $\binom{3}{2} = 3$ ways to get x^2y. Similarly, there are $\binom{3}{1} = 3$ ways to get xy^2 and only one way to get y^3. Hence

$$(x+y)^3 = \binom{3}{3}x^3 + \binom{3}{2}x^2y + \binom{3}{1}xy^2 + \binom{3}{0}y^3 = x^3 + 3x^2y + 3xy^2 + y^3.$$

The general case is exactly the same. When multiplied out, the product

$$(x+y)^n = (x+y)\cdot(x+y)\cdot(x+y)\cdots(x+y) \tag{4.3}$$

is a sum of terms $x^n, x^{n-1}y, \ldots, xy^{n-1}, y^n$. We get copies of x^jy^{n-j} by choosing x from any j of the factors in (4.3) and then taking y from the other $n-j$ factors. Thus we get $\binom{n}{j}$ copies of x^jy^{n-j}. Summing over the possible values of j gives (4.1), which completes the proof of the binomial theorem. □

Example 4.11. We use the binomial theorem to compute

$$\begin{aligned}(2t+3)^4 &= \binom{4}{4}(2t)^4 + \binom{4}{3}(2t)^3\cdot 3 + \binom{4}{2}(2t)^2\cdot 3^2 + \binom{4}{1}2t\cdot 3^3 + \binom{4}{0}3^4\\ &= 16t^4 + 4\cdot 8t^3\cdot 3 + 6\cdot 4t^2\cdot 9 + 4\cdot 2t\cdot 27 + 81\\ &= 16t^4 + 96t^3 + 216t^2 + 216t + 81.\end{aligned}$$

4.2 The Vigenère cipher

The simple substitution ciphers that we studied in Section 1.1 are examples of *monoalphabetic ciphers*, since every plaintext letter is encrypted using only one cipher alphabet. As cryptanalytic methods became more sophisticated in Renaissance Italy, correspondingly more sophisticated ciphers were invented (although it seems that they were seldom used in practice). Consider how much more difficult a task is faced by the cryptanalyst if every plaintext letter is encrypted using a different ciphertext alphabet. This ideal resurfaces in modern cryptography in the form of the one-time pad, which we discuss in Section 4.6, but in this section we discuss a less complicated *polyalphabetic cipher* called the Vigenère cipher[4] dating back to the 16th century.

[4]This cipher is named after Blaise de Vigenère (1523–1596), whose 1586 book *Traicté des Chiffres* describes the known ciphers of his time. These include polyalphabetic ciphers such as the "Vigenère cipher," which according to [58] Vigenère did not invent, and an ingenious autokey system (see Exercise 4.19), which he did.

The Vigenère cipher works by using different shift ciphers to encrypt different letters. In order to decide how far to shift each letter, Bob and Alice first agree on a keyword or phrase. Bob then uses the letters of the keyword, one by one, to determine how far to shift each successive plaintext letter. If the keyword letter is `a`, there is no shift, if the keyword letter is `b`, he shifts by 1, if the keyword letter is `c`, he shifts by 2, and so on. An example illustrates the process:

Example 4.12. Suppose that the keyword is `dog` and the plaintext is `yellow`. The first letter of the keyword is `d`, which gives a shift of 3, so Bob shifts the first plaintext letter `y` forward by 3, which gives the ciphertext letter `b`. (Remember that `a` follows `z`.) The second letter of the keyword is `o`, which gives a shift of 14, so Bob shifts the second plaintext letter `e` forward by 14, which gives the ciphertext letter `s`. The third letter of the keyword is `g`, which gives a shift of 6, so Bob shifts the third plaintext letter `l` forward by 6, which gives the ciphertext letter `r`.

Bob has run out of keyword letters, so what does he do now? He simply starts again with the first letter of the keyword. The first letter of the keyword is `d`, which again gives a shift of 3, so Bob shifts the fourth plaintext letter `l` forward by 3, which gives the ciphertext letter `o`. Then the second keyword letter `o` tells him to shift the fifth plaintext letter `o` forward by 14, giving the ciphertext letter `c`, and finally the third keyword letter `g` tells him to shift the sixth plaintext letter `w` forward by 6, giving the ciphertext letter `c`.

In conclusion, Bob has encrypted the plaintext `yellow` using the keyword `dog` and obtained the ciphertext `bsrocc`.

Even this simple example illustrates two important characteristics of the Vigenère cipher. First, the repeated letters `ll` in the plaintext lead to non-identical letters `ro` in the ciphertext, and second, the repeated letters `cc` in the ciphertext correspond to different letters `ow` of the plaintext. Thus a straightforward frequency analysis as we used to cryptanalyze simple substitution ciphers (Section 1.1.1) is not going to work for the Vigenère cipher.

A useful tool for doing Vigenère encryption and decryption, at least if no computer is available (as was typically the case in the 16^{th} century!), is the so-called *Vigenère tableau* illustrated in Figure 4.1. The Vigenère tableau consists of 26 alphabets arranged in a square, with each alphabet shifted one further than the alphabet to its left. In order to use a given keyword letter to encrypt a given plaintext letter, Bob finds the plaintext letter in the top row and the keyword letter in the first column. He then looks for the letter in the tableau lying below the plaintext letter and to the right of the keyword letter. That is, he locates the encrypted letter at the intersection of the row beginning with the keyword letter and the column with the plaintext letter on top.

For example, if the keyword letter is `d` and the plaintext letter is `y`, Bob looks in the fourth row (which is the one that starts with `d`) and in the next

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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

- Find the plaintext letter in the top row.
- Find the keyword letter in the first column.
- The ciphertext letter lies below the plaintext letter and to the right of the keyword letter.

Table 4.1: The Vigenère Tableau

to last column (which is the one headed by y). This row and column intersect at the letter b, so the corresponding ciphertext letter is b.

Decryption is just as easy. Alice uses the row containing the keyword letter and looks in that row for the ciphertext letter. Then the top of that column is the plaintext letter. For example, if the keyword letter is g and the ciphertext letter is r, Alice looks in the row starting with g until she finds r and then she moves to the top of that column to find the plaintext letter l.

Example 4.13. We illustrate the use of the Vigenère tableau by encrypting the plaintext message

```
The rain in Spain stays mainly in the plain,
```

using the keyword flamingo. Since the key word has eight letters, the first step is to split the plaintext into eight-letter blocks,

```
theraini | nspainst | aysmainl | yinthepl | ain.
```

Next we write the keyword beneath each block of plaintext, where for convenience we label lines $\mathcal{P}$, $\mathcal{K}$, and $\mathcal{C}$ to indicate, respectively, the plaintext, the keyword, and the ciphertext.

$\mathcal{P}$	t h e r a i n i	n s p a i n s t	a y s m a i n l	y i n t h e p l	a i n
$\mathcal{K}$	f l a m i n g o	f l a m i n g o	f l a m i n g o	f l a m i n g o	f l a

Finally, we encrypt each letter using the Vigenère tableau. The initial plaintext letter t and initial keyword letter f combine in the Vigenère tableau to yield the ciphertext letter y, the second plaintext letter h and second keyword letter l combine in the Vigenère tableau to yield the ciphertext letter s, and so on. Continuing in this fashion, we complete the encryption process.

$\mathcal{P}$	t h e r a i n i	n s p a i n s t	a y s m a i n l	y i n t h e p l	a i n
$\mathcal{K}$	f l a m i n g o	f l a m i n g o	f l a m i n g o	f l a m i n g o	f l a
$\mathcal{C}$	y s e d i v t w	s d p m q a y h	f j s y i v t z	d t n f p r v z	f t n

Splitting the ciphertext into convenient blocks of five letters each, we are ready to transmit our encrypted message

```
ysedi  vtwsd  pmqay  hfjsy  ivtzd  tnfpr  vzftn.
```

Remark 4.14. As we already pointed out, the same plaintext letter in a Vigenère cipher is represented in the ciphertext by many different letters. However, if the keyword is short, there will be a tendency for repetitive parts of the plaintext to end up aligned at the same point in the keyword, in which case they will be identically enciphered. This occurs in Example 4.13, where the ain in rain and in mainly are encrypted using the same three keyword letters ing, so they yield the same ciphertext letters ivt. This repetition in the ciphertext, which appears separated by 16 letters, suggests that the keyword has length dividing 16. Of course, not every occurrence of ain in the

plaintext yields the same ciphertext. It is only when two occurrences line up with the same part of the keyword that repetition occurs.

In the next section we develop the idea of using ciphertext repetitions to guess the length of the keyword, but here we simply want to make the point that short keywords are less secure than long keywords.[5] On the other hand, Bob and Alice find it easier to remember a short keyword than a long one. We thus see the beginnings of the eternal struggle in practical (as opposed to purely theoretical) cryptography, namely the battle between

$$\textbf{Efficiency (and ease of use)} \longleftarrow \text{versus} \longrightarrow \textbf{Security}.$$

As a further illustration of this dichotomy, we consider ways in which Bob and Alice might make their Vigenère-type cipher more secure. They can certainly make Eve's job harder by mixing up the letters in the first row of their Vigenère tableau and then rotating this "mixed alphabet" in the subsequent rows. Unfortunately, a mixed alphabet makes encryption and decryption more cumbersome, plus it means that Bob and Alice must remember (or write down for safekeeping!) not only their keyword, but also the mixed alphabet. And if they want to be even more secure, they can use different randomly mixed alphabets in every row of their Vigenère tableau. But if they do that, then they will certainly need to keep a written copy of the tableau, which is a serious security risk.

4.2.1 Cryptanalysis of the Vigenère cipher: theory

At various times in history it has been claimed that Vigenère-type ciphers, especially with mixed alphabets, are "unbreakable." In fact, nothing could be further from the truth. If Eve knows Bob and Alice, she may be able to guess part of the keyword and proceed from there. (How many people do you know who use some variation of their name and birthday as an Internet password?) But even without lucky guesses, elementary statistical methods developed in the 19^{th} century allow for a straightforward cryptanalysis of Vigenère-type ciphers. In the interest of simplicity, we stick with the original Vigenère, i.e., we do not allow mixed alphabets in the tableau.

You may wonder why we take the time to cryptanalyze the Vigenère cipher, since no one these days uses the Vigenère for secure communications. The answer is that our exposition is designed principally to introduce you to the use of statistical tools in cryptanalysis. This builds on and extends the elementary application of frequency tables as we used them in Section 1.1.1 to cryptanalyze simple substitution ciphers. In this section we describe the theoretical tools used to cryptanalyze the Vigenère, and in the next section we apply those tools to decrypt a sample ciphertext. If at any point you find

[5] More typically one uses a key phrase consisting of several words, but for simplicity we use the term "keyword" to cover both single keywords and longer key phrases.

that the theory in this section becomes confusing, it may help to turn to Section 4.2.2 and see how the theory is applied in practice.

The first goal in cryptanalyzing a Vigenère cipher is to find the length of the keyword, which is sometimes called the *blocksize* or the *period*. We already saw in Remark 4.14 how this might be accomplished by looking for repeated fragments in the ciphertext. The point is that certain plaintext fragments such as `the` occur quite frequently, while other plaintext fragments such as `ugw` occur infrequently or not at all. Among the many occurrences of the letters `the` in the plaintext, a certain percentage of them will line up with exactly the same part of the keyword.

This leads to the *Kasiski method*, first described by a German military officer named Friedrich Kasiski in his book *Die Geheimschriften und die Dechiffrir-kunst*[6] published in 1863. One looks for repeated fragments within the ciphertext and compiles a list of the distances that separate the repetitions. The key length is likely to divide many of these distances. Of course, a certain number of repetitions will occur by pure chance, but these are random, while the ones coming from repeated plaintext fragments are always divisible by the key length. It is generally not hard to pick out the key length from this data.

There is another method of guessing the key length that works with individual letters, rather than with fragments consisting of several letters. The underlying idea can be traced all the way back to the frequency table of English letters (Table 1.3), which shows that some letters are more likely to occur than others. Suppose now that you are presented with a ciphertext encrypted using a Vigenère cipher and that you guess that it was encrypted using a keyword of length 5. This means that every fifth letter was encrypted using the same rotation, so if you pull out every fifth letter and form them into a string, this entire string was encrypted using a single substitution cipher. Hence the string's letter frequencies should look more or less as they do in English, with some letters much more frequent and some much less frequent. And the same will be true of the string consisting of the 2^{nd}, 7^{th}, 12^{th},... letters of the ciphertext, and so on. On the other hand, if you guessed wrong and the key length is not five, then the string consisting of every fifth letter should be more or less random, so its letter frequencies should look different from the frequencies in English.

How can we quantify the following two statements so as to be able to distinguish between them?

String 1 has letter frequencies similar to those in Table 1.3. (4.4)

String 2 has letter frequencies that look more or less random. (4.5)

One method is to use the following device.

[6] *Cryptography and the Art of Decryption.*

Definition. Let $\mathbf{s} = c_1c_2c_3\cdots c_n$ be a string of n alphabetic characters. The *index of coincidence* of $\mathbf{s}$, denoted by IndCo($\mathbf{s}$), is the probability that two randomly chosen characters in the string $\mathbf{s}$ are identical.

We are going to derive a formula for the index of coincidence. It is convenient to identify the letters a,...,z with the numbers $0, 1, \ldots, 25$ respectively. For each value $i = 0, 1, 2, \ldots, 25$, let F_i be the frequency with which letter i appears in the string $\mathbf{s}$. For example, if the letter h appears 23 times in the string $\mathbf{s}$, then $F_7 = 23$, since h = 7 in our labeling of the alphabet.

For each i, there are $\binom{F_i}{2} = \frac{F_i(F_i-1)}{2}$ ways to select two instances of the i^{th} letter of the alphabet from $\mathbf{s}$, so the total number of ways to get a repeated letter is the sum of $\frac{F_i(F_i-1)}{2}$ for $i = 0, 1, \ldots, 25$. On the other hand, there are $\binom{n}{2} = \frac{n(n-1)}{2}$ ways to select two arbitrary characters from $\mathbf{s}$. The probability of selecting two identical letters is the total number of ways to choose two identical letters divided by the total number of ways to choose any two letters. That is,

$$\text{IndCo}(\mathbf{s}) = \frac{1}{n(n-1)} \sum_{i=0}^{25} F_i(F_i - 1). \tag{4.6}$$

Example 4.15. Let $\mathbf{s}$ be the string

$$\mathbf{s} = \text{“A bird in hand is worth two in the bush.”}$$

Ignoring the spaces between words, $\mathbf{s}$ consists of 30 characters. The following table counts the frequencies of each letter that appears at least once:

	A	B	D	E	H	I	N	O	R	S	T	U	W
i	0	1	3	4	7	8	13	14	17	18	19	20	22
F_i	2	2	2	1	4	4	3	2	2	2	3	1	2

Then the index of coincidence of $\mathbf{s}$, as given by (4.6), is

$$\text{IndCo}(\mathbf{s}) = \frac{1}{30 \cdot 29}(2{\cdot}1+2{\cdot}1+2{\cdot}1+4{\cdot}3+4{\cdot}3+3{\cdot}2+\cdots+3{\cdot}2+2{\cdot}1) \approx 0.0575.$$

We return to our two statements (4.4) and (4.5). Suppose first that the string $\mathbf{s}$ consists of random characters. Then the probability that $c_i = c_j$ is exactly $\frac{1}{26}$, so we would expect IndCo($\mathbf{s}$) $\approx \frac{1}{26} \approx 0.0385$. On the other hand, if $\mathbf{s}$ consists of English text, then we would expect the relative frequencies to be as in Table 1.3. So for example, if $\mathbf{s}$ consists of 10,000 characters, we would expect approximately 815 A's, approximately 144 B's, approximately 276 C's, and so on. Thus the index of coincidence for a string of English text should be approximately

$$\frac{815 \cdot 814 + 144 \cdot 143 + 276 \cdot 275 + \cdots + 8 \cdot 7}{10000 \cdot 9999} \approx 0.0685.$$

The disparity between 0.0385 and 0.0685, as small as it may seem, provides the means to distinguish between Statement 4.4 and Statement 4.5. More precisely:

If IndCo($\mathbf{s}$) ≈ 0.068, then $\mathbf{s}$ looks like simple substitution English. (4.7)

If IndCo($\mathbf{s}$) ≈ 0.038, then $\mathbf{s}$ looks like random letters. (4.8)

Of course, the value of IndCo($\mathbf{s}$) will tend to fluctuate, especially if $\mathbf{s}$ is fairly short. But the moral of (4.7) and (4.8) is that larger values of IndCo($\mathbf{s}$) make it more likely that $\mathbf{s}$ is English encrypted with some sort of simple substitution, while smaller values of IndCo($\mathbf{s}$) make it more likely that $\mathbf{s}$ is random.

Now suppose that Eve intercepts a message $\mathbf{s}$ that she believes was encrypted using a Vigenère cipher and wants to check whether the keyword has length k. Her first step is to break the string $\mathbf{s}$ into k pieces $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k$, where $\mathbf{s}_1$ consists of every k^{th} letter starting from the first letter, $\mathbf{s}_2$ consists of every k^{th} letter starting from the second letter, and so on. In mathematical terms, if we write $\mathbf{s} = c_1c_2c_3\ldots c_n$, then

$$\mathbf{s}_i = c_ic_{i+k}c_{i+2k}c_{i+3k}\ldots.$$

Notice that if Eve's guess is correct and the keyword has length k, then each $\mathbf{s}_i$ consists of characters that were encrypted using the same shift amount, so although they do not decrypt to form actual words (remember that $\mathbf{s}_i$ is every k^{th} letter of the text), the pattern of their letter frequencies will look like English. On the other hand, if Eve's guess is incorrect, then the $\mathbf{s}_i$ strings will be more or less random.

Thus for each k, Eve computes IndCo($\mathbf{s}_i$) for $i = 1, 2, \ldots, k$ and checks whether these numbers are closer to 0.068 or closer to 0.038. She does this for $k = 3, 4, 5, \ldots$ until she finds a value of k for which the average value of IndCo($\mathbf{s}_1$), IndCo($\mathbf{s}_2$), $\ldots$, IndCo($\mathbf{s}_k$) is large, say greater than 0.06. Then this k is probably the correct blocksize.

We assume now that Eve has used the Kasiski test or the index of coincidence test to determine that the keyword has length k. That's a good start, but she's still quite far from her goal of finding the plaintext. The next step is to compare the strings $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k$ to one another. The tool she uses to compare different strings is called the mutual index of coincidence. The general idea is that each of the k strings has been encrypted using a different shift cipher. If the string $\mathbf{s}_i$ is shifted by β_i and the string $\mathbf{s}_j$ is shifted by β_j, then one would expect the frequencies of $\mathbf{s}_i$ to best match those of $\mathbf{s}_j$ when the symbols in $\mathbf{s}_i$ are shifted by an additional amount

$$\sigma \equiv \beta_j - \beta_i \pmod{26}.$$

This leads to the following useful definition.

Definition. Let

$$\mathbf{s} = c_1c_2c_3\ldots c_n \qquad \text{and} \qquad \mathbf{t} = d_1d_2d_3\ldots d_m$$

be strings of alphabetic characters. The *mutual index of coincidence* of $\mathbf{s}$ and $\mathbf{t}$, denoted by $\mathrm{MutIndCo}(\mathbf{s},\mathbf{t})$, is the probability that a randomly chosen character from $\mathbf{s}$ and a randomly chosen character from $\mathbf{t}$ will be the same.

If we let $F_i(\mathbf{s})$ denote the number of times the i^{th} letter of the alphabet appears in the string $\mathbf{s}$, and similarly for $F_i(\mathbf{t})$, then the probability of choosing the i^{th} letter from both is the product of the probabilities $\frac{F_i(\mathbf{s})}{n}$ and $\frac{F_i(\mathbf{t})}{m}$. In order to obtain a formula for the mutual index of coincidence of $\mathbf{s}$ and $\mathbf{t}$, we add these probabilities over all possible letters,

$$\mathrm{MutIndCo}(\mathbf{s},\mathbf{t}) = \frac{1}{nm}\sum_{i=0}^{25} F_i(\mathbf{s})F_i(\mathbf{t}). \tag{4.9}$$

Example 4.16. Let $\mathbf{s}$ and $\mathbf{t}$ be the strings

$\mathbf{s}$ = "A bird in hand is worth two in the bush,"
$\mathbf{t}$ = "A stitch in time saves nine."

Using formula (4.9) to compute the mutual index of coincidence of $\mathbf{s}$ and $\mathbf{t}$ yields $\mathrm{MutIndCo}(\mathbf{s},\mathbf{t}) = 0.0773$.

The mutual index of coincidence has very similar properties to the index of coincidence. For example, there are analogues of the two statements (4.7) and (4.8). The value of $\mathrm{MutIndCo}(\mathbf{s},\mathbf{t})$ can be used to confirm that a guessed shift amount is correct. Thus if two strings $\mathbf{s}$ and $\mathbf{t}$ are encrypted using the *same* simple substitution cipher, then $\mathrm{MutIndCo}(\mathbf{s},\mathbf{t})$ tends to be large, because of the uneven frequency with which letters appear. On the other hand, if $\mathbf{s}$ and $\mathbf{t}$ are encrypted using *different* substitution ciphers, then they have no relation to one another, and the mutual index of coincidence $\mathrm{MutIndCo}(\mathbf{s},\mathbf{t})$ will be much smaller.

We return now to Eve's attack on a Vigenère cipher. She knows the key length k and has split the ciphertext into k blocks, $\mathbf{s}_1,\mathbf{s}_2,\ldots,\mathbf{s}_k$, as usual. The characters in each block have been encrypted using the same shift amount, say

$$\beta_i = \text{Amount that block } \mathbf{s}_i \text{ has been shifted.}$$

Eve's next step is to compare $\mathbf{s}_i$ with the string obtained by shifting the characters in $\mathbf{s}_j$ by different amounts. As a notational convenience, we write

$$\mathbf{s}_j + \sigma = \begin{pmatrix}\text{The string } \mathbf{s}_j \text{ with every character}\\ \text{shifted } \sigma \text{ spots down the alphabet.}\end{pmatrix}$$

Suppose that σ happens to equal $\beta_i - \beta_j$. Then $\mathbf{s}_j+\sigma$ has been shifted a total of $\beta_j + \sigma = \beta_i$ from the plaintext, so $\mathbf{s}_j + \sigma$ and $\mathbf{s}_i$ have been encrypted using the same shift amount. Hence, as noted above, their mutual index of coincidence will be fairly large. On the other hand, if σ is not equal to $\beta_i - \beta_j$, then $\mathbf{s}_j + \sigma$

and $\mathbf{s}_i$ have been encrypted using different shift amounts, so $\mathrm{MutIndCo}(\mathbf{s},\mathbf{t})$ will tend to be small.

To put this concept into action, Eve computes all of the mutual indices of coincidence

$$\mathrm{MutIndCo}(\mathbf{s}_i, \mathbf{s}_j + \sigma) \qquad \text{for } 1 \le i < j \le k \text{ and } 0 \le \sigma \le 25.$$

Scanning the list of values, she picks out the ones that are large, say larger than 0.065. Each large value of $\mathrm{MutIndCo}(\mathbf{s}_i, \mathbf{s}_j + \sigma)$ makes it likely that

$$\beta_i - \beta_j \equiv \sigma \pmod{26}. \tag{4.10}$$

(Note that (4.10) is only a congruence modulo 26, since a shift of 26 is the same as a shift of 0.) The leads to a system of equations of the form (4.10) for the variables $\beta_1, \ldots, \beta_k$. In practice, some of these equations will be spurious, but after a certain amount of trial and error, Eve will end up with values $\gamma_2, \ldots, \gamma_k$ satisfying

$$\beta_2 = \beta_1 + \gamma_2, \quad \beta_3 = \beta_1 + \gamma_3, \quad \beta_4 = \beta_1 + \gamma_4, \quad \ldots, \quad \beta_k = \beta_1 + \gamma_k.$$

Thus if the keyword happens to start with `A`, then the second letter of the keyword would be `A` shifted by γ_2, the third letter of the keyword would be `A` shifted by γ_3, and so on. Similarly, if the keyword happens to start with `B`, then its second letter would be `B` shifted by γ_2, its third letter would be `A` shifted by γ_3, etc. So all that Eve needs to do is try each of the 26 possible starting letters and decrypt the message using each of the 26 corresponding keywords. Looking at the first few characters of the 26 putative plaintexts, it is easy for her to pick out the correct one.

Remark 4.17. We make one final remark before doing an example. We noted earlier that among the many occurrences of the letters `the` in the plaintext, a certain percentage of them will line up with exactly the same part of the keyword. It turns out that these repeated encryptions occur much more frequently than one might guess. This is an example of the "birthday paradox," which says that the probability of getting a match (e.g. of trigrams or birthdays or colors) is quite high. We discuss the birthday paradox and some of its many applications to cryptography in Section 4.4.

4.2.2 Cryptanalysis of the Vigenère cipher: practice

In this section we illustrate how to cryptanalyze a Vigenère ciphertext by decrypting the message given in Table 4.2.

We begin by applying the Kasiski test. A list of repeated trigrams is given in Table 4.3, together with their location within the ciphertext and the number of letters that separates them. Most of the differences in the last column are divisible by 7, and 7 is the largest number with this property, so we guess that the keyword length is 7.

```
zpgdl rjlaj kpylx zpyyg lrjgd lrzhz qyjzq repvm swrzy rigzh
zvreg kwivs saolt nliuw oldie aqewf iiykh bjowr hdogc qhkwa
jyagg emisr zqoqh oavlk bjofr ylvps rtgiu avmsw lzgms evwpc
dmjsv jqbrn klpcf iowhv kxjbj pmfkr qthtk ozrgq ihbmq sbivd
ardym qmpbu nivxm tzwqv gefjh ucbor vwpcd xuwft qmoow jipds
fluqm oeavl jgqea lrkti wvext vkrrg xani
```

Table 4.2: A Vigenère ciphertext to cryptanalyze

Trigram	Appears at places	Difference
avl	117 and 258	$141 = 3 \cdot 47$
bjo	86 and 121	$35 = 5 \cdot 7$
dlr	4 and 25	$21 = 3 \cdot 7$
gdl	3 and 24	$16 = 2^4$
lrj	5 and 21	$98 = 2 \cdot 7^2$
msw	40 and 138	$84 = 2^2 \cdot 3 \cdot 7$
pcd	149 and 233	$13 = 13$
qmo	241 and 254	$98 = 2 \cdot 7^2$
vms	39 and 137	$84 = 2^2 \cdot 3 \cdot 7$
vwp	147 and 231	$84 = 2^2 \cdot 3 \cdot 7$
wpc	148 and 232	$21 = 3 \cdot 7$
zhz	28 and 49	$21 = 3 \cdot 7$

Table 4.3: Repeated trigrams in the ciphertext given in Table 4.2

Although the Kasiski test shows that the period is probably 7, we also apply the index of coincidence test in order to to illustrate how it works. Table 4.4 lists the indices of coincidence for various choices of key length and the average index of coincidence for each key length. We see from Table 4.4 that key length 7 has far higher average index of coincidence than the other potential key lengths, which confirms the conclusion from the Kasiski test.

Now that Eve knows that the key length is 7, she compares the blocks with one another as described in Section 4.2.1. She first breaks the ciphertext into seven blocks by taking every seventh letter. (Notice how the first seven letters of the ciphertext run down the first column, the second seven down the second column, and so on.)

$\mathbf{s}_1 =$ zlxrhrrhwloehdweoklilwvlhphqbynwhwfjulrxx
$\mathbf{s}_2 =$ pazjzezzitlwboamqbvuzpjpvmtiimiquptiqjkta
$\mathbf{s}_3 =$ gjpgqpyvvndfjgjihjpagcqckfkhvqvvccqpmgtvn
$\mathbf{s}_4 =$ dkydyvrrsliiocysoosvmdbfxkobdmxgbdmdoqiki
$\mathbf{s}_5 =$ lpyljmiesieiwqarafrmsmrijrzmapmeoxoseewr
$\mathbf{s}_6 =$ rygrzsggauayrhgzvrtsejnobqrqrbtfruofaavr
$\mathbf{s}_7 =$ jllzqwzkowqkhkgqlygwvskwjtgsduzjvwwlvleg

Key Length	Average Index	Individual Indices of Coincidence
4	0.038	0.034, 0.042, 0.039, 0.035
5	0.037	0.038, 0.039, 0.043, 0.027, 0.036
6	0.036	0.038, 0.038, 0.039, 0.038, 0.032, 0.033
7	0.062	0.062, 0.057, 0.065, 0.059, 0.060, 0.064, 0.064
8	0.038	0.037, 0.029, 0.038, 0.030, 0.034, 0.057, 0.040, 0.039
9	0.037	0.032, 0.036, 0.028, 0.030, 0.026, 0.032, 0.045, 0.047, 0.056

Table 4.4: Indices of coincidence of Table 4.2 for various key lengths

She then compares the i^{th} block $\mathbf{s}_i$ to the j^{th} block shifted by σ, which we denote by $\mathbf{s}_j + \sigma$, taking successively $\sigma = 0, 1, 2, \ldots, 25$. Table 4.5 gives a complete list of the 546 mutual indices of coincidence

$$\text{MutIndCo}(\mathbf{s}_i, \mathbf{s}_j + \sigma) \qquad \text{for } 1 \le i < j \le 7 \text{ and } 0 \le \sigma \le 25.$$

In Table 4.5, the entry in the row corresponding to (i, j) and the column corresponding to the shift σ is equal to

$$\text{MutIndCo}(\mathbf{s}_i, \mathbf{s}_j + \sigma) = \text{MutIndCo}(\text{Block } \mathbf{s}_i, \text{Block } \mathbf{s}_j \text{ shifted by } \sigma). \quad (4.11)$$

If this quantity is large, it suggests that $\mathbf{s}_j$ has been shifted σ further than $\mathbf{s}_i$. As in Section 4.2.1 we let

$$\beta_i = \text{Amount that the block } \mathbf{s}_i \text{ has been shifted.}$$

Then a large value for (4.11) makes it likely that

$$\beta_i - \beta_j = \sigma. \quad (4.12)$$

We have underlined the large values (those greater than 0.065) in Table 4.5 and compiled them, with the associated shift relation (4.12), in Table 4.6.

Eve's next step is to solve the system of linear equations appearing in the final column of Table 4.6, keeping in mind that all values are modulo 26, since a shift of 26 is the same as no shift at all. Notice that there are 10 equations for the six variables $\beta_1, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7$. (Unfortunately, β_2 does not appear, so we'll deal with it later). In general, a system of 10 equations in 6 variables has no solutions,[7] but in this case a little bit of algebra shows that not only is there a solution, there is actually one solution for each value of β_1. In other words, the full set of solutions is obtained by expressing each of the variables $\beta_3, \ldots, \beta_7$ in terms of β_1:

$$\beta_3 = \beta_1 + 25, \quad \beta_4 = \beta_1 + 7, \quad \beta_5 = \beta_1 + 1, \quad \beta_6 = \beta_1 + 10, \quad \beta_7 = \beta_1 + 15. \quad (4.13)$$

[7]We were a little lucky in that every relation in Table 4.6 is correct. Sometimes there are erroneous relations, but it is not hard to eliminate them with some trial and error.

Blocks		Shift Amount												
i	j	0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	.025	.034	.045	.049	.025	.032	.037	.042	.049	.031	.032	.037	.043
1	3	.023	**.067**	.055	.022	.034	.049	.036	.040	.040	.046	.025	.031	.046
1	4	.032	.041	.027	.040	.045	.037	.045	.028	.049	.042	.042	.030	.039
1	5	.043	.021	.031	.052	.027	.049	.037	.050	.033	.033	.035	.044	.030
1	6	.037	.036	.030	.037	.037	.055	.046	.038	.035	.031	.032	.037	.032
1	7	.054	.063	.034	.030	.034	.040	.035	.032	.042	.025	.019	.061	.054
2	3	.041	.029	.036	.041	.045	.038	.060	.031	.020	.045	.056	.029	.030
2	4	.028	.043	.042	.032	.032	.047	.035	.048	.037	.040	.028	.051	.037
2	5	.047	.037	.032	.044	.059	.029	.017	.044	.060	.034	.037	.046	.039
2	6	.033	.035	.052	.040	.032	.031	.031	.029	.055	.052	.043	.028	.023
2	7	.038	.037	.035	.046	.046	.054	.037	.018	.029	.052	.041	.026	.037
3	4	.029	.039	.033	.048	.044	.043	.030	.051	.033	.034	.034	.040	.038
3	5	.021	.041	.041	.037	.051	.035	.036	.038	.025	.043	.034	.039	.036
3	6	.037	.034	.042	.034	.051	.029	.027	.041	.034	.040	.037	.046	.036
3	7	.046	.023	.028	.040	.031	.040	.045	.039	.020	.030	**.069**	.042	.037
4	5	.041	.033	.041	.038	.036	.031	.056	.032	.026	.034	.049	.029	.054
4	6	.035	.037	.032	.039	.041	.033	.032	.039	.042	.031	.049	.039	.058
4	7	.031	.032	.046	.038	.039	.042	.033	.056	.046	.027	.027	.036	.036
5	6	.048	.036	.026	.031	.033	.039	.037	.027	.037	.045	.032	.040	.041
5	7	.030	.051	.043	.031	.034	.041	.048	.032	.053	.037	.024	.029	.045
6	7	.032	.033	.030	.038	.032	.035	.047	.050	.049	.033	.057	.050	.021

Blocks		Shift Amount												
i	j	13	14	15	16	17	18	19	20	21	22	23	24	25
1	2	.034	.052	.037	.030	.037	.054	.021	.018	.052	.052	.043	.042	.046
1	3	.031	.037	.038	.050	.039	.040	.026	.037	.044	.043	.023	.045	.032
1	4	.039	.040	.032	.041	.028	.019	**.071**	.038	.040	.034	.045	.026	.052
1	5	.042	.032	.038	.037	.032	.045	.045	.033	.041	.043	.035	.028	.063
1	6	.040	.030	.028	**.071**	.051	.033	.036	.047	.029	.037	.046	.041	.027
1	7	.040	.032	.049	.037	.035	.035	.039	.023	.043	.035	.041	.042	.027
2	3	.054	.040	.028	.031	.039	.033	.052	.046	.037	.026	.028	.036	.048
2	4	.047	.034	.027	.038	.047	.042	.026	.038	.029	.046	.040	.061	.025
2	5	.034	.026	.035	.038	.048	.035	.033	.032	.040	.041	.045	.033	.036
2	6	.033	.034	.036	.036	.048	.040	.041	.049	.058	.028	.021	.043	.049
2	7	.042	.037	.041	.059	.031	.027	.043	.046	.028	.021	.044	.048	.040
3	4	.037	.045	.033	.028	.029	**.073**	.026	.040	.040	.026	.043	.042	.043
3	5	.035	.029	.036	.044	.055	.034	.033	.046	.041	.024	.041	**.067**	.037
3	6	.023	.043	**.074**	.047	.033	.043	.030	.026	.042	.045	.032	.035	.040
3	7	.035	.035	.035	.028	.048	.033	.035	.041	.038	.052	.038	.029	.062
4	5	.032	.041	.036	.032	.046	.035	.039	.042	.038	.034	.043	.036	.048
4	6	.034	.034	.036	.029	.043	.037	.039	.036	.039	.033	**.066**	.037	.028
4	7	.043	.032	.039	.034	.029	**.071**	.037	.039	.030	.044	.037	.030	.041
5	6	.052	.035	.019	.036	.063	.045	.030	.039	.049	.029	.036	.052	.041
5	7	.040	.031	.034	.052	.026	.034	.051	.044	.041	.039	.034	.046	.029
6	7	.029	.035	.039	.032	.028	.039	.026	.036	**.069**	.052	.035	.034	.038

Table 4.5: Mutual indices of coincidence of Table 4.2 for shifted blocks

What should Eve do about β_2? She could just ignore it for now, but instead she picks out the largest values in Table 4.5 that relate to block 2 and uses those. The largest such values are $(i, j) = (2, 3)$ with shift 6 and index 0.060 and $(i, j) = (2, 4)$ with shift 24 and index 0.061, which give the relations

$$\beta_2 - \beta_3 = 6 \qquad \text{and} \qquad \beta_2 - \beta_4 = 24.$$

Substituting in from (4.13), these both yield $\beta_2 = \beta_1 + 5$, and the fact that they give the same value gives Eve confidence that they are correct.

To summarize, Eve now knows that however much the first block $\mathbf{s}_1$ is rotated, blocks $\mathbf{s}_2, \mathbf{s}_3, \ldots, \mathbf{s}_7$ are rotated, respectively, 5, 25, 7, 1, 10, and 15 steps further than $\mathbf{s}_1$. So for example, if $\mathbf{s}_1$ is not rotated at all (i.e., if $\beta_1 = 0$ and the first letter of the keyword is `A`), then the full keyword is `AFZHBKP`. Eve uses the keyword `AFZHBKP` to decrypt the first few blocks of the ciphertext, finding the "plaintext"

`zkhwkhulvkdoowxuqrxwwrehwkhkhurripbrzqolihruzkhwkh`.

i	j	Shift	MutIndCo	Shift Relation
1	3	1	0.067	$\beta_1 - \beta_3 = 1$
3	7	10	0.069	$\beta_3 - \beta_7 = 10$
1	4	19	0.071	$\beta_1 - \beta_4 = 19$
1	6	16	0.071	$\beta_1 - \beta_6 = 16$
3	4	18	0.073	$\beta_3 - \beta_4 = 18$
3	5	24	0.067	$\beta_3 - \beta_5 = 24$
3	6	15	0.074	$\beta_3 - \beta_6 = 15$
4	6	23	0.066	$\beta_4 - \beta_6 = 23$
4	7	18	0.071	$\beta_4 - \beta_7 = 18$
6	7	21	0.069	$\beta_6 - \beta_7 = 21$

Table 4.6: Large indices of coincidence and shift relations

Shift	Keyword	Decrypted Text
0	AFZHBKP	zkhwkhulvkdoowxuqrxwwrehwkhkhurripbrzqolih
1	BGAICLQ	yjgvjgtkujcnnvwtpqwvvqdgvjgjgtqqhoaqypnkhg
2	CHBJDMR	xifuifsjtibmmuvsopvuupcfuififsppgnzpxomjgf
3	DICKENS	whetherishallturnouttobetheheroofmyownlife
4	EJDLFOT	vgdsgdqhrgzkkstqmntssnadsgdgdqnnelxnvmkhed
5	FKEMGPU	ufcrfcpgqfyjjrsplmsrrmzcrfcfcpmmdkwmuljgdc
6	GLFNHQV	tebqebofpexiiqroklrqqlybqebebollcjvltkifcb
7	HMGOIRW	sdapdaneodwhhpqnjkqppkxapdadankkbiuksjheba
8	INHPJSX	rczoczmdncvggopmijpoojwzoczczmjjahtjrigdaz
⋮	⋮	⋮

Table 4.7: Decryption of Table 4.2 using shifts of the keyword AFZHBKP

That doesn't look good! So next she tries $\beta_1 = 1$ and a keyword starting with the letter B. Continuing in this fashion, she need only check the 26 possibilities for β_1. The results are listed in Table 4.7.

Taking $\beta_1 = 3$ yields the keyword DICKENS and an acceptable plaintext. Completing the decryption using this keyword and supplying the appropriate word breaks, punctuation, and capitalization, Eve recovers the full plaintext:

> Whether I shall turn out to be the hero of my own life, or whether that station will be held by anybody else, these pages must show. To begin my life with the beginning of my life, I record that I was born (as I have been informed and believe) on a Friday, at twelve o'clock at night. It was remarked that the clock began to strike, and I began to cry, simultaneously.[8]

[8] *David Copperfield*, 1850, Charles Dickens