BREAKING COMPOUND CIPHERS

Breaking Compound Ciphers

madness

August, 2022

This is a short (ahem) introduction to breaking compound ciphers. All ciphers are classical, and none are modern. It is best if the reader (you) is familiar with the ciphers that we use, and might even know how to break some of them. This is not the first document you should read if you are just starting to explore classical cryptography.

What is a compound cipher?

Simply put, a *compound cipher* is one that can be broken up, or *factored*, into a sequence of simpler ciphers. Sometimes a familiar cipher turns out to be factorable; other times, we deliberately chain together two ciphers to make a more difficult cipher. To break a compound cipher, we need to break each of the simpler ciphers that make it up.

Occasionally, one of those factor ciphers might be *keyless*, i.e., always encrypt the same way because a key is not used. For example, think of Bacon's biliteral cipher; in it, each letter is always replaced with the same five-character sequence. Having such a factor makes the task of breaking the full cipher easier, as we shall see in some examples.

It also often happens that a cipher's decryptor is the same sort of operation as its encryptor. For example, a monoalphabetic substitution cipher (a cipher in which each letter is replaced by another letter according to a key table) is decrypted with a substitution cipher using a different key. We might say that the key for decryption is the *inverse* of the key for encryption, or that the substitution cipher used to decrypt is the *inverse* of the substitution cipher used to encrypt. For a counter-example, consider again Bacon's biliteral cipher; its encryption and decryption operations are very different. Whether a factor cipher is its own inverse will sometimes have a bearing on how we break the full cipher.

When more than one component cipher is used, then decryption must be done in *reverse order*. So if a plaintext p becomes a ciphertext c by the action of some ciphers Q, R, S, T like this:

$$c = T(S(R(Q(p))))$$

then decryption is done like this:

$$p = Q^{-1}(R^{-1}(S^{-1}(T^{-1}(c))))$$

Notice that in this notation, action of the ciphers proceeds from right to left. The decryptor of a cipher Q is written as Q^{-1}, since it is the inverse operation of Q.

In the next section we discuss whether the order of the component ciphers is strictly enforced, of if there are cases when the order can be changed.

Commutivity

The most important property of the factors that make up the full cipher is whether they must operate in a certain order, or whether the order does not matter. If the order matters, then decryption of the full cipher requires that the factor ciphers be used in reverse order. If the order is irrelevant, then we can break the factors in whatever order is most convenient for us. In that case, we say that the factors *commute*. For example, consider a Caesar shift cipher and a columnar transposition cipher. It makes no difference whether a text is encrypted with the Caesar cipher first or with the transposition cipher first; the ciphertext will be identical for either choice. Mathematicians represent this graphically with a commutation diagram, like this:

plaintext p	$\rightarrow$	Caesar shift K	$\rightarrow$	intermediate text A
$\downarrow$				$\downarrow$
transposition T				transposition T
$\downarrow$				$\downarrow$
intermediate text B	$\rightarrow$	Caesar shift K	$\rightarrow$	ciphertext c

(*K* is for *Kaiser*, so as not to be confused with *c* for c*iphertext*.) The two intermediate texts *A* and *B* are different. If we prefer to write it as an equation:

$$c = K(T(p)) = T(K(p))$$

When a cipher maps from one character set to another, as in the Polybius square or Morse code, then we know that it will not commute with other ciphers.

Sometimes, factor ciphers will not truly commute, but almost. In these cases, if we switch the order of the ciphers, then the keys must change. As an example, consider a permutation cipher Q_K followed by a Vigenère cipher V_L. The subscripts denote the keys. If the length of the permutation key *K* is the same as the Vigenère key *L*, then we can switch the order of the two ciphers, but the keys must change.

$$c = \mathrm{V}_L(Q_K(p)) = Q_K(V_{L'}(p))$$

Here, L' (you have to look closely to see the prime) denotes the modified Vigenère key. The permutation key does not, in this example, need to be modified. We leave it as an exercise for the reader to determine whether

$$L' = Q_K(L)$$

or

$$L' = Q_K^{-1}(L)$$

It is always true that a monoalphabetic substitution and a transposition cipher will commute.

Some familiar ciphers that are compound

Affine cipher

Believe it or not, the affine cipher is compound. It is made up of a multiplication cipher followed by an addition cipher. These two factors do not commute; they must be undone in the proper order when we decrypt a ciphertext. The encryption function can be written

$$c_i = a\,p_i + b \mod 26$$

But to correctly decrypt, we must use

$$p_i = a^{-1}\,(c_i - b) \mod 26$$

Notice how the parentheses enforce the order of operations, which is reverse of that used in encryption.

Polybius cipher

What we usually think of the Polybius cipher involves filling a 5×5 square with a mixed alphabet, then replacing each letter of the plaintext with the two coordinates of that letter in the square. But this can be factored into two pieces. In the first, a monoalphabetic substitution is applied. In the second, each letter is replaced with the coordinates in Polybius square containing an *unmixed* alphabet (this factor is keyless). The key for the substitution factor is the inverse of the mixed alphabet used in the original version of the cipher.

The two factors involve different kinds of operation and they do not commute. However, knowing that the cipher can be factored will help us break it, as we shall see below.

ADFGX (and ADFGVX)

The German ADFGX (and its 6×6 version the ADFGVX) consists of a Polybius cipher followed by a columnar transposition cipher. As we just saw, the Polybius factor can in turn be factored into a substitution cipher and an unscrambled Polybius cipher. None of these factors commute, so when we break this cipher, we will need to do each factor in the correct order.

Quagmire ciphers

The cipher known as quagmire 1 is a monoalphabetic substitution cipher followed by a Vigenère cipher. The quagmire 2 is a Vigenère cipher followed by a monoalphabetic substitution. The quagmire 3 is a substitution followed by a Vigenère, followed by the inverse of the substitution. In the quagmire 4, we have one substitution, then a Vigenère, then a different substitution.

In all four cases, the factors do not commute. Also, you should not expect the keys for the factors to be the keys you might expect by looking at the tableaux used to encrypt by hand, i.e., not the same as the keywords you would use.

Some component ciphers

These are some component ciphers that we will consider. The list is not exhaustive. For example, there are many more transposition ciphers than the two included here.

Monoalphabetic substitution

This is the simple substitution cipher with which we are all familiar. It is called *monoalphabetic* to distinguish it from the periodic polyalphabetic ciphers like the Vigenère. The Caesar shift and affine cipher are two special cases of the monoalphabetic substitution, and we will discuss them further below.

We all know how to break a monoalphabetic substitution by trying partial keys and guessing at what words appear as we try to fill in the remainder of the key. But with a computer we can speed this up with an algorithm published by Jakobsen in 1995 in *Cryptologia*. His is a hill-climbing method in which a key is modified by swapping two letters in it. If the resulting decryption is better than the one using the old key, then the new key is kept. Modifications of the new key are tried until a better decryption is found and a newer key is obtained. This continues until no better decryption can be found. How good a decryption is is determined by counting two-letter combinations in the plaintext, and comparing their frequencies to those of typical English text (or we can use three-letter or four-letter combinations).

It is sometimes useful to be able to invert the key for a monoalphabetic substitution. The inverse key $\overline{K}$ is the key for a substitution that is the decryptor of the original substitution (whose key is K):

$$p = S_K^{-1}(c) = S_{\overline{K}}(c)$$

There is an easy way to find the inverse key. Suppose our key is

```
KEYWORDABCFGHIJLMNPQSTUVXZ
```

We write this under the unmixed alphabet, keeping in mind that the substitution acts by replacing a letter on the top row by the one below it.

```
abcdefghijklmnopqrstuvwxyz
KEYWORDABCFGHIJLMNPQSTUVXZ
```

Then we rearrange the columns until the bottom row is no longer mixed.

```
hijgbklmnoapqrestfuvwxdycz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

The new top row is the inverse key.

Caesar shift cipher

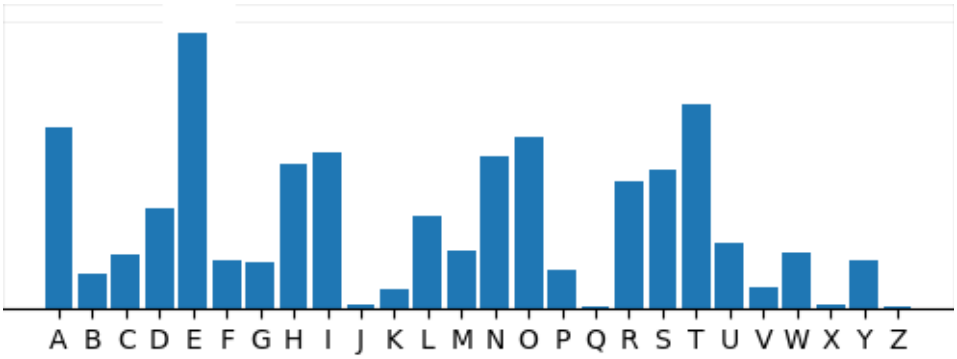
We all know how this cipher works. Each letter is shifted by the same amount along the alphabet, with wrap-around. For example, if the key is 18, we can envision the action of the cipher like the following, in which plaintext letters from the top row are replaced with the letter below them in the bottom row:

```
                        abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
```

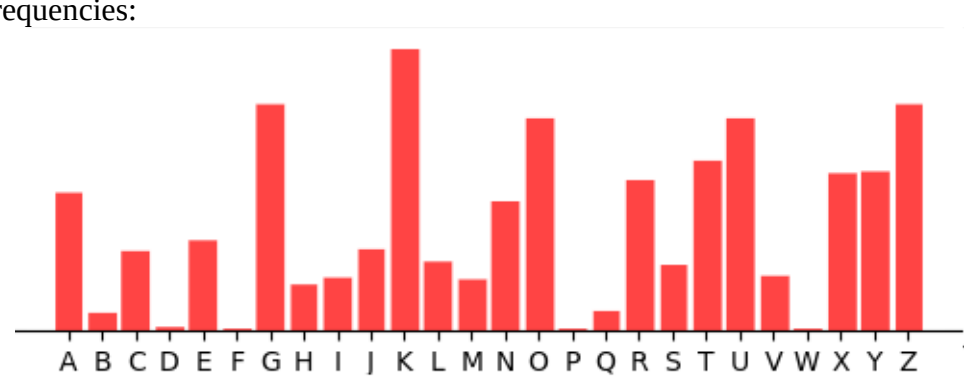
The inverse of key K is

$$\overline{K} = 26 - K$$

We all know that a very simple way to break a Caesar cipher is to try each of the 25 possible keys ($K = 0$ is irrelevant) and look for a good decryption. However, if a text has been scrambled by a transposition cipher, then this method cannot work. So let's explore a method of breaking the Caesar cipher even when the ciphertext has been scrambled. We will do this with letter frequencies. We first need to know the frequencies of the twenty-six letters from typical English text. Here are the frequencies that I found from some British novels. The vertical units are arbitrary.

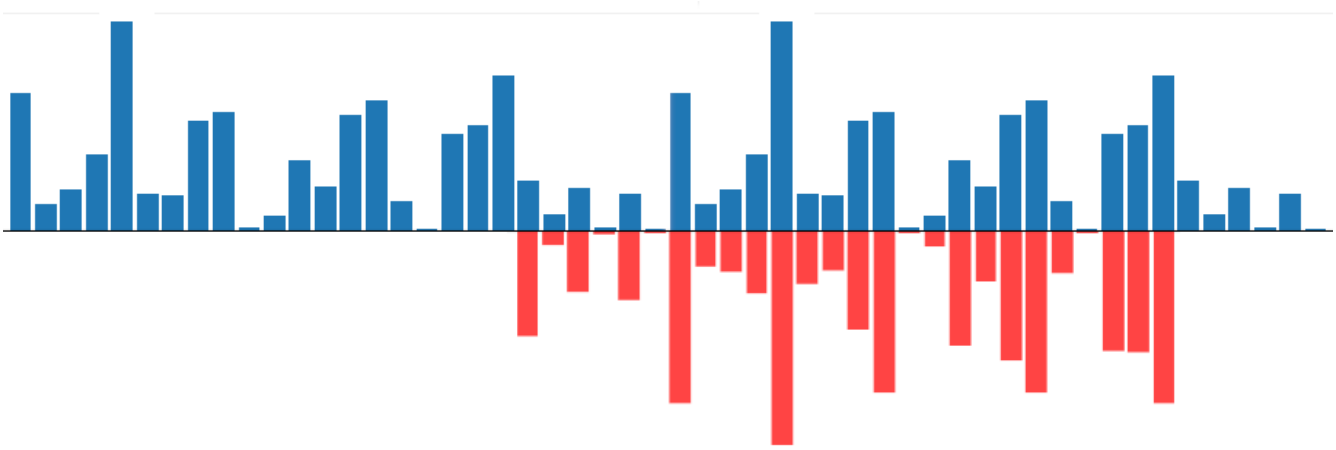


Now let's consider a ciphertext; we will take as our example challenge 1B from the 2020 competition. Here are its frequencies:



To make a comparison easier, we can make two copies of the English frequencies and put the ciphertext frequencies directly below:

At first glance, the largest peak in blue and in red are six steps apart. The fact that the key is 6 is quite obvious when we shift the red distribution left by six steps:



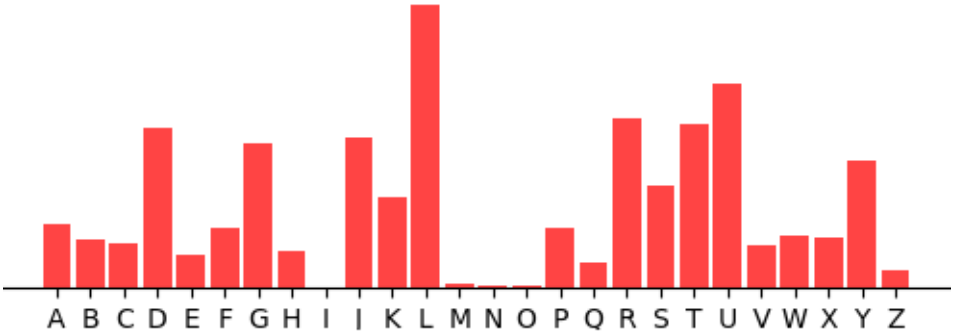
From the excellent match that we see above, we now know that the key for the Caesar cipher in this example is 6.

If you wish to automate this method, you can use the χ^2 method of determining the best fit, or you can do what I prefer: maximizing the sum of the products of the blue bars each with the red bar below it.

Affine cipher

Can we break an affine even when the text has been scrambled? Yes. Two methods come to mind: solving the modular equations, and checking all possible keys.

For an example of solving the modular equations, let's use the ciphertext from challenge 2B from the 2021 competition. Here are its frequencies:



When we look at the English frequencies in the graph earlier, the highest peak is for E = 4, and the second is for A = 0 or T = 19. In the graph for the ciphertext, the highest is L = 11 and the second is U = 20. Let's try a couple of choices and see if we can recover the key algebraically. For E → L and A → U, we have

$$\begin{aligned} 11 &= 4\,a + b \\ 20 &= 0\,a + b \end{aligned}$$

Remember that the arithmetic must be done modulo 26. Unfortunately, there is no solution. For E → L and T → U, we have

$$\begin{aligned} 11 &= 4\,a + b \\ 20 &= 19\,a + b \end{aligned}$$

This set of equations does have a solution: $a = 11$, $b = 19$, and this is, indeed, the key to the affine cipher. Had it not been, we would have to try maybe a handful of other possibilities.

To break an affine by trying all possible keys, even when the text is scrambled, we can do something similar to the method used for the Caesar shift. In this case, however, we do not shift the frequencies. Instead, we must shuffle them according to the affine cipher and then compare. There are 311 possible keys, since *a* can take values 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25, and *b* can take any value from 0 to 25 (*a* must be invertible modulo 26, so must not be divisible by 2 or 13, which are the factors of 26). Since the key (1, 0) is the identity operation, there are 311 possibilities to consider. For each possibility, we start with the frequencies of typical English, rearrange them under the action of the affine cipher, then compare to the frequencies of the ciphertext. The best match indicates that we have found the key. Again, it is up to you whether to use the χ^2 method of determining the best fit, or my way of maximizing the sum of the square of the frequencies.

Vigenère cipher

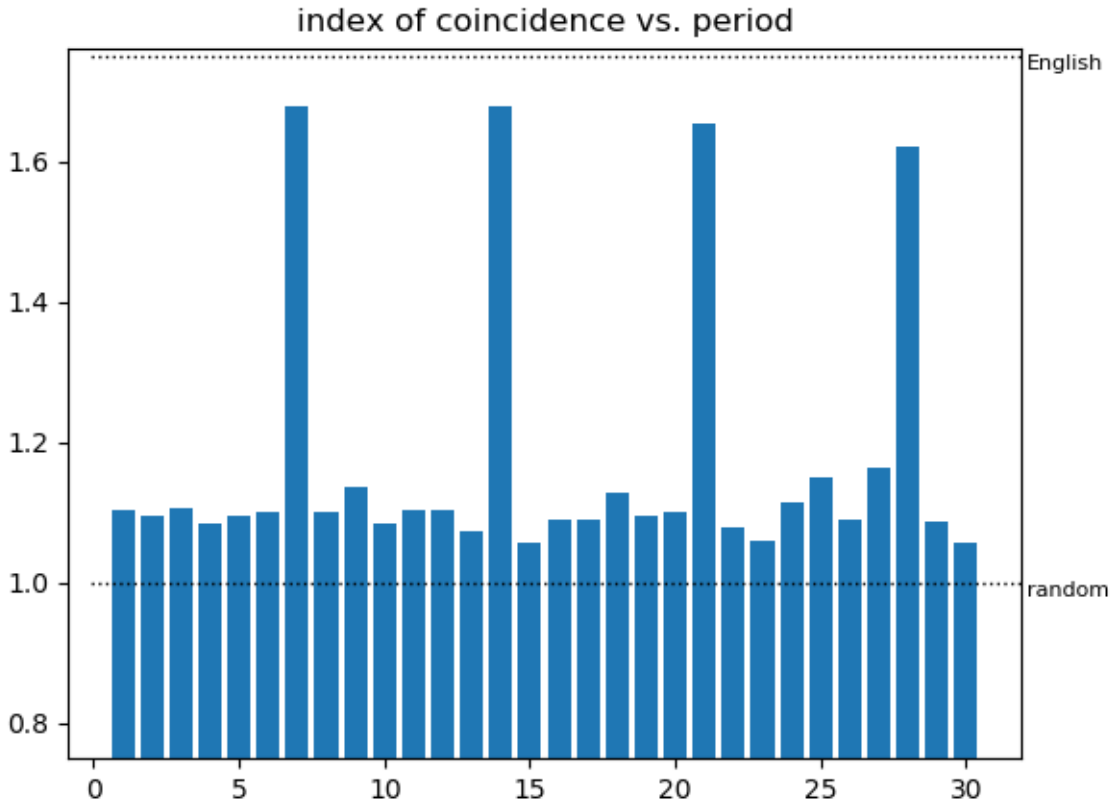
I think we all know what a Vigenère cipher is, but just to be clear: A Vigenère cipher is a periodic collection of Caesar shift ciphers. In other words, There is some set of *m* Caesar shift ciphers, and they are applied one at a time as we encrypt each of the letters in the plaintext. When we get to the end of the set of Caesar shifts, we go back to the beginning and continue encrypting (hence the use of the word *periodic*).

There are a few methods for finding the period of the cipher, which is the same as the number *m* of Caesar shifts. One is the Kasisky method, which involves finding groups of characters that reoccur in the ciphertext. The distances between repeating groups are multiples of the period. So the period is likely to be the GCD (greatest common divisor) of those distances.

Another way to find the period is with the index of coincidence (IoC), which measures the likelihood that any two characters of a text are the same. A concise formula for the IoC is

$$I = 26 \, \Sigma_{i=A}^{Z} \, n_i \, (n_i - 1) \, / \, N \, (N - 1)$$

where n_i are the counts of the letters in the text, and N is the total number of characters. Notice that I use a normalization factor of 26 which many cryptanalysts do not use. With this normalization, a random text has an IoC close to 1, while English text is close to 1.7. If you don't like this formula, then you can just as easily use a simplified version which is 26 times the sum of the squares of the frequencies of the letters in the text, divided by the square of the sum of the frequencies (that latter sum is, of course, 1, so you can ignore it). To use the IoC to find the period of the cipher, we cut the ciphertext into *m* slices, where each slice contains every m^{th} letter. Then we find the IoC for each slice and average them. We do this for various choices of *m*. The smallest *m* with an average IoC close to 1.7 is our period. For example, below we see an example of the graph of the IoC versus the number of slices *m*. The period for this example is 7.



Once we know the period of the cipher, we can break each of the Caesar shift ciphers using the method described above on each of the *m* slices of the ciphertext.

Keyless Polybius square

For this cipher, we cram the alphabet into a 5×5 matrix, called a *Polybius square*. Unfortunately, the English alphabet has more than 5^2 letters, so we may have to jettison one of them. Usually, we will put ‘I’ and ‘J’ in the same spot. We then add labels to the rows and columns. The action of the cipher is to replace each letter in the plaintext with its coordinates in the Polybius square.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

The decryption method should be obvious: We simply replace each pair of numbers in the ciphertext with the letter in the corresponding position in the square.

Since this cipher maps symbols in one set to symbols in a different set, i.e., {A, ..., Z} → {1, 2, 3, 4, 5}, it cannot commute with other ciphers.

Morse code

In Morse code, letters are replaced by codewords made of dots (·) and dashes (−). Codewords are separated by spaces, which are often denoted with crosses (×). I won’t bore you or take up space by presenting the table of codewords. This “cipher” has no key, and decipherment is simply to replace each codeword with the letter that it represents.

Again, since this “cipher” maps symbols in one set to symbols in a different set, i.e., {A, ..., Z} → {·, −, ×}, it cannot commute with other ciphers.

Permutation cipher

In this cipher, the plaintext is divided into blocks of equal length. Each block is permuted (scrambled) in the same manner. Decryption is also a permutation cipher, in which the key is the inverse permutation of the encrypting cipher.

If you like, you can think of this cipher as writing the plaintext into a matrix, then shuffling the columns of the matrix, and finally reading the ciphertext out one row at a time.

Now, what we need is a way to break such a cipher when the letters of the ciphertext have all been changed by a substitution cipher. The index of coincidence will not help us, since it only deals with the counts or frequencies of letters, and not with their positions. We need some measure that takes into account groupings of letters. So we introduce the *index of coincidence for digrams* (IoC2). Rather than counting individual letters (monograms), we will be counting pairs of letters. Here is a formula:

$$I_2 = 26^2 \Sigma_{i=\mathrm{AA}}^{\mathrm{ZZ}} n_i (n_i - 1) / N (N - 1)$$

The sum runs over all pairs AA, AB, AC, ..., ZY, ZZ. The n_i are the counts of the pairs, and N is the total number of pairs. If you don't like this formula, then notice that for a long ciphertext, the 1's can be ignored, and the formula reduces to the sum of the frequencies squared. And if you like, we can use counts instead of frequencies, since we are just going to maximize the value anyway. This simplified IoC2 can be calculated like this:

1. Make a list of all the digrams (two-letter sequences) in the text.
2. Count the occurrences of each digram.
3. Square the count for each digram in the list.
4. Add all the squares. The result is the sIoCd.

You will want to use a computer for this, because we need to do this calculation every time we try a new key for the transposition cipher. However you decide to calculate the IoC2, since English is a repetitive encoding of ideas, the IoC2 will be large. So we want to maximize its value. To do so, we use a hill-climbing technique: Try a key and calculate the IoC2 for the decryption. Then modify the key by either swapping two elements in it, or by shifting it right or left, with wrap-around. If the IoC2 of the new decryption is higher, then we keep the new key and try modifying it. This continues until we can't do any better. At that point, it is very likely that we have the correct key.

Columnar transposition

In this cipher, we write the plaintext into a matrix, shuffle the columns, then read the ciphertext out one column at a time. In this case, the decryption process is not a version of the encryptor. Instead, the ciphertext must be written into the matrix one *column* at a time, and the plaintext read out one row at a time.

Once again, what we need is a way to break such a cipher when the letters of the ciphertext have all been changed by a substitution cipher. We will use the same hill-climbing technique as outlined above for the permutation cipher, but with the decryptor of the columnar transposition cipher. We will have to guess at the length of the key (the number of columns). But this is not such a bad thing; it is easy to just try several key lengths until we find a solution.

Homophonic substitution

In a homophonic substitution, a set of symbols is mapped to a larger set in such a way that we have more than one choice for the replacement of a plaintext symbol. Breaking such a cipher is done the same way as a monoalphabetic substitution. However, we have map more ciphertext symbols to plaintext symbols. Nevertheless, the modifications to the method are straightforward.

Combining similar ciphers

This is just an aside to explain that when two similar ciphers are combined, the result is often not a new cipher.

When one monoalphabetic substitution cipher is followed by a second monoalphabetic substitution, the result is the same as a single monoalphabetic substitution:

$$c = S_2(S_1(p)) = S_3(p)$$

The key of the resulting substitution is

$$K_3 = S_2(K_1)$$

When two Caesar shift ciphers are combined, the result is a single Caesar cipher with key

$$K_3 = K_1 + K_2$$

When two affine ciphers are combined, the result is also an affine cipher. We can see this by looking at the action of the two ciphers on a plaintext letter whose number is x to give a ciphertext letter with number y:

$$\begin{aligned} y &= a_2 (a_1 x + b_1) + b_2 \\ &= (a_2 a_1) x + (a_2 b_1 + b_2) \end{aligned}$$

So the key of the resulting cipher is

$$\begin{aligned} a_3 &= a_1 a_2 \\ b_3 &= a_2 b_1 + b_2 \end{aligned}$$

where all operations are done modulo 26.

When two Vigenère ciphers are combined, the result is a Vigenère cipher whose keylength is the LCM (least common multiple) of the original two keylengths. The key of the resulting cipher is the encryption of several copies of one of the original keys by a Vigenère using the other original key:

$$K_3 = V_2(K_1 K_1 K_1 ...) = V_1(K_2 K_2 K_2 ...)$$

Breaking Polybius cipher

As mentioned above, a Polybius cipher that uses a square containing a shuffled alphabet can be factored into a monoalphabetic substitution cipher followed by a keyless Polybius cipher (i.e., one that uses an unshuffled alphabet in its square). This makes the full cipher no harder to break than a simple substitution cipher. First, undo the keyless Polybius cipher. What remains is the substitution cipher, which you should know how to break.

Something to think about: Is the mixed alphabet in the square the same as the key of the substitution cipher? Is it the inverse key?

Breaking Caesar shift cipher and transposition cipher

A monoalphabetic substitution and any transposition cipher commute, and a Caesar cipher is a type of monoalphabetic substitution. So we can attack the two factor ciphers in either order. While it is possible to break the transposition first, it is far easier to break the Caesar shift first, using the method based on letter frequencies explained above. The transposition can then be broken by whatever method is appropriate for whatever type of transposition we have.

Breaking affine cipher and transposition cipher

Can we apply something similar to the case of an affine and a transposition cipher? Yes. The affine cipher is also a type of monoalphabetic substitution, so it commutes with the transposition. So wecan break the affine cipher first. We have two methods explained above for doing so. The transposition can then be broken by whatever method is appropriate for whatever type of transposition we have.

Breaking monoalphabetic substitution and transposition cipher

In the case of a generic monoalphabetic substitution cipher, we do not have a method of breaking it when the ciphertext is scrambled. Therefore, we must break the transposition cipher first. For transposition ciphers of the permutation and columnar type, we have a hill-climbing attack outlined above. After deciphering the transposition cipher, what remains is to break the substitution cipher, and we have a hill-climbing method for doing that as well.

Breaking ADFGX

Breaking an ADFGX cipher is very much like the combination of monoalphabetic substitution with columnar transposition. The difference comes from adding a keyless Polybius cipher between them. This additional factor keeps the substitution and transposition from commuting. So we are required to break the columnar transposition first. Typically, we will guess the key length for the columnar transposition. If we guess wrong, we guess again until we succeed. For a given key length, we must maximize the IoC2. It is calculated for each transposition key by deciphering the transposition and the keyless Polybius and finding the IoC2 for the resulting text. Once the columnar transposition is cracked, we are able to work on the remaining substitution cipher.

Breaking quagmire 1 cipher

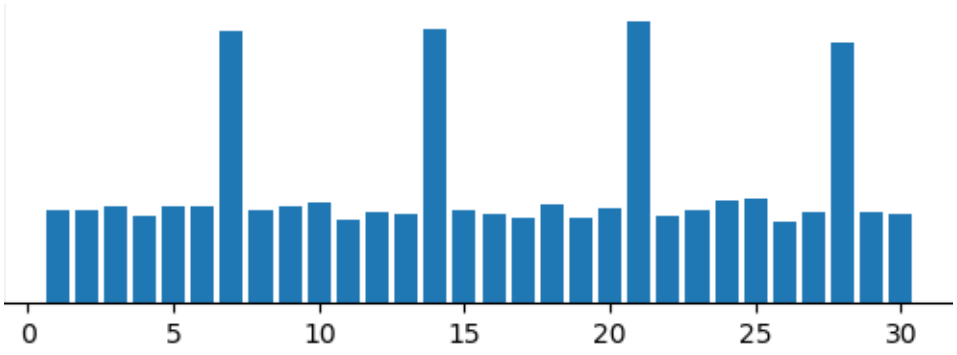
The quagmire 1 cipher is a monoalphabetic substitution followed by a Vigenère cipher. They do not commute, so we must attack the Vigenère first. Because of the underlying substitution, we cannot use the same attack that we use on an uncompounded Vigenère (breaking it as a series of Caesar shifts). However, the technique is similar. Finding the period is done the same way as with a Vigenère. Once the period is known, we cannot find the shifts of the letter frequencies compared to typical English text. Instead, we can only find the relative shifts between the various slices of the ciphertext.

To clarify the method, let’s work through an example. Here is a ciphertext:

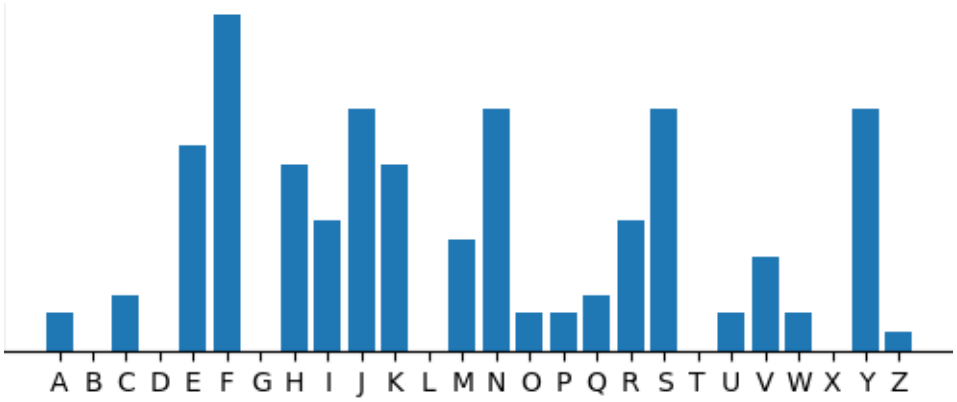
```
RQWGXSZKMILKJKKHGUHJSCBHBCRSYYIIJXMKSHBMHNVQSOERSYYIIJANZQCK
BTXJNCFOPMQJLUVWQSMWNKRSIVTSCOUFSKONARKHDCERAJMPJSTPMIDQKRSJ
```

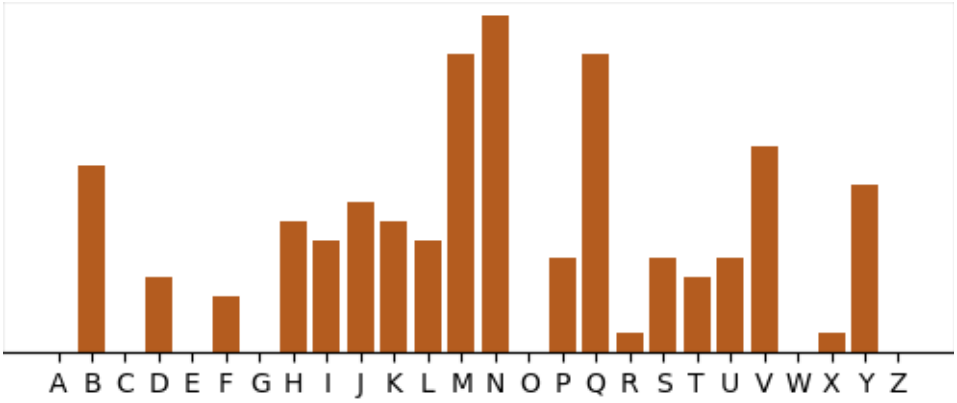
```
NCFOPMYNGIZBRYQLRKRKFJMNCOGNNMBJRPSTPFGOHSBQFCPMIQIETVSHQHNK
HSSLYNCJZFNCJZOOYMIRSWMNLIAGAOEMPJSTPMIDQKRSANAGNTAHNSOERNPB
IEEWGYBLJSWQJIMACHGRVWLHXVVFFGSWAYKDFPEEFHHGESZNBIOFAVYQLRKR
RFUIVKHNVQHNPQYFKPOVEAHLMBEIVEBIFPOPANWKJWSEPAUCEVJUQDNAVPQK
FOWQSLFFOTHNMDADHZKMIAGAOESLIGTIUDIAFBVEFDCVSSEMKJFEVRQWGPRU
KYLIZSZNYIDJONUQGACPPIMDNPESQNQDHQQYPQMOSUCHIOVWQRVWJZWTEIYU
OEDVYPJVTAIVWGPRKFJWFHNRMQGRKRNHIHHKTUKHTJFBISPKDCSZKHTYCPSE
MPCBYMFDFJFTRRVTHDSKFJWNCSSJJWFOWMNDLAKEAYBQIKHZNBIOFSZNYIDL
AOOXKJFHUSTPFJLSHNEFCESJVMUZHRNYPJVTAINFENTHNNWRHVUYVONHQWSQ
FFLODWJHJBHMKKOJFBPJULHCJVMBDLGAVWPIDKSRSRSGXTKJULEHNCVVKIKQ
PSNIDFSZNNOJVASHTPJHIKFKHNKBMIJWFZRSJNJGEDGSMPOEDRRQAFHDVJUL
BLXSYVDDHVEFMWNCSRFHLIPWHNMPJSOPFMDDZAVMIDQKRYENXJJLSJNCFOPM
KKGUHBQSYIMBWSJSIDCYGOVLDLWKFJSDCJNRQCAKTUMFWGPRSEMPCBJZFDIU
IWYSKFOEZRFSQDFSZNYIDVXOEVDCVDAHLKYSTPHQCEPHWFKMJBSAHTDNZWGY
NQEGAVFPGRHQSHLWOSFUCBHBTFVIMIIRBPQ
```

You can find the period with a Kasiski examination. For example, SZN occurs five time in the text. The separations between them are 140, 168, 133, and 189. The only factor they have in common is 7, so we are somewhat confident that the period of the cipher is 7. We can also look at the index of coincidence for slices of every nth letter:

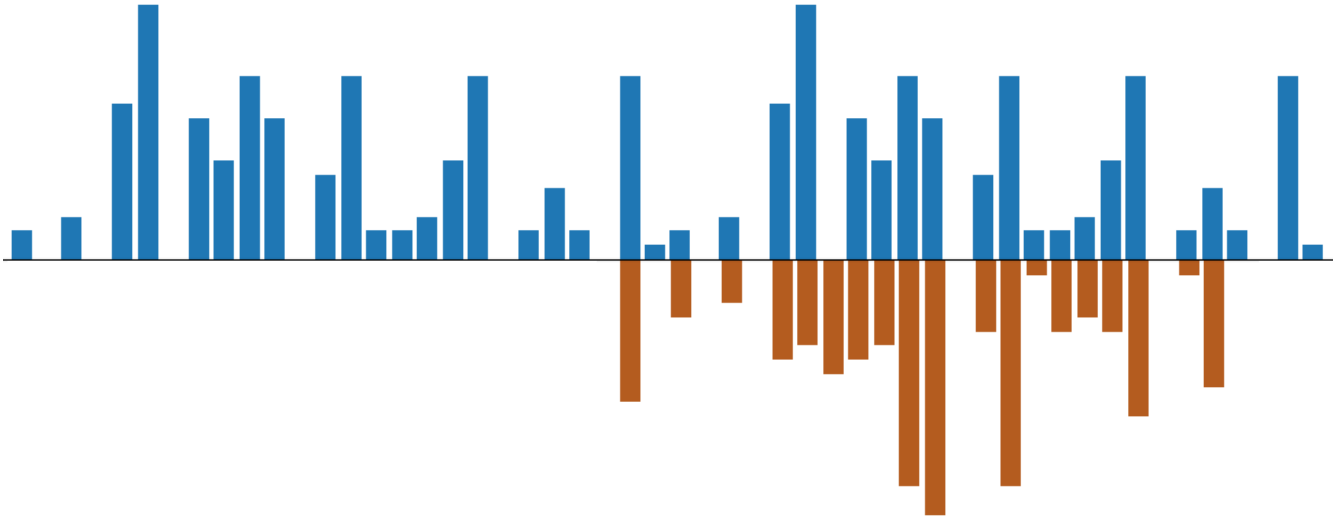


This seems convincing. Now that we know the period, we divide the ciphertext into slices of every 7th letter, starting on the first, on the second, on the third, ... For each slice, we must find the frequencies of the letters. Here are the distributions of the first two slices:





Like we did for the Caesar cipher, we can find the relative shift between these two slices. It's not pretty, but the best fit is to shift the second slice three steps to the left relative to the first slice:



In a similar fashion, we find that the seven shifts, relative to the first slice, are:

slice 1:	0	(of course)
slice 2:	3	
slice 3:	24	
slice 4:	22	
slice 5:	23	
slice 6:	9	
slice 7:	8	

Once we unshift each slice and put the text back together, we obtain this:

```
RNYKAJRKJKPNACKEIYKAKCYJFFIKYVKMMOEKPJFPYFVNUSHIKYVKMMRFZNEO
EKPJKEJRGEQGNYYNISJYRNIKISVWFFMFPMSQRJKEFGHISJJRNVKHMFFUNIKJ
KEJRGEYKIMCSJYNNVNICFGORFFYNKOFMIHSQRJJFZSYSJFGEINKIWMKHNJRN
YKSIARFARFKENCFGYJKVVNENIKEJRGEJRNVKHMFFUNIKAKCKQKSHKUSHIFPY
KIHNYYYNNVNIJFOEFYYRSYPKONVCHKVNSYHFJSVWFEJKHJRNYKSIRNYNNVNI
JFRKZNYFVNJRSHQFHRSYVSHIOFHZNEYKJSFHAKYOMNKEMCYFVNJRSHQRNPNM
```

```
JRNISIHJRKZNJFEGYRKJKEJRGEPNMJKAUAKEISNECFGYJKEJMNIVNRNYKSIM
KVNMCJRNVKHMFFUNIEFGHIJFRSVKQKSHKHIYMSQRJMCEKSYNIRSYNCNLEFAY
RVVVVRNYKSISYKSICFGYJKEJMNIVNIFHFJLNKMKEVNISASMMHFJRKEVCFGKE
JRGEPEFAHNIKJRSVLGJCFGYRFJKJGYJRNENANENVSYYSMNYRNYKSIJRNVKHO
RGOUMNIYMSQRJMCKHKGJFVKJSOYCYJNVRNYKSIKHIQKZNKYVKMMYSQRKHOSN
HJOFVWGJNEYEKHQNISHJRNLFANMYFPJRNWMKHNJJSOUKAKCJRNIKEUVSMMNH
HSKKHIJRNKQNYRKHQRNKZCFHJRNSEIGYJCIKJKLKHUYSJRSHUJRNCJKUNJRN
FOOKYSFHKMWFJYRFJJFENMSNZNJRNVFHFJFHCRNMFFUNIQEKZNMCKJKEJRGE
KHIYKSISVKQENKJPKHFPYOSNHONCFGUHFAFRNEENKMMCYKSIKEJRGEARFAKY
LNQSHHSHQJFPSHIJRNVKHYOGESFGYUSHIMCVKHHNEISYOFHONEJSHQFRCNYY
KSIJRNFMIVKHKHIYSVWMCYJFWWNIJKMUSHQ
```

This text can now be broken as a single monoalphabetic substitution cipher. Once we do so, we find that the Vigenère key is `ADYWXJI`, and the substitution key is `KLOINPQRSDUMVHFWBEYJGZATCX`. These look like gibberish, and they almost are. However, we have forgotten that we only found the shifts *relative* to the first slice. If we assume that the first slice is also shifted, and run through the 26 possibilities, we find more meaningful keys: `KNIGHTS WJYQROUNDTABLECFGHIXKMPZSV`. The keywords used to generate the ciphertext were, in fact, `KNIGHTS` and `ROUNDTABLE`.

Breaking fractionated Morse

You wish I would reveal my secrets.

Breaking a Pollux cipher

See note above, for fractionated Morse.

Breaking a Polybius cipher followed by homophonic substitution

This sounds interesting. I suspect that the method for Pollux could be modified to handle such a cipher.

Identification of compound ciphers

Determining that you have a ciphertext that was encrypted with a compound cipher can be difficult. Here are some ideas:

Suppose you have

1. Single letter frequencies do not match English;
2. IoC is close to that of English text;
3. IoC2 is too small compared to English text;
4. IoC does not show a definite period, as for the Vigenère cipher (see above).

Then it is likely that you have a substitution cipher and a transposition cipher.