An Introduction to Traditional Cryptography and Cryptanalysis for Amateurs

Chris Spackman

August 2003

Table of Contents

1 Introduction

This book is about traditional cryptography – the use and analysis of traditional, pre-computer ciphers. We will start with simple substitution and introduce progressively more advanced ciphers.

Conventions Used in this Book

Ciphers and codes are different. However, I am not going to deal with codes in this book at all, so if I occasionally use the words `code', `encode', or `decode' please understand them to refer to ciphers, not codes. For most laymen there is no difference between ciphers and codes so I shall use them interchangably.

Plaintext will be in `typewriter` text.

Ciphertext will be in SMALL CAPITALS text.

Warning: Randomness

Random is a loaded word in cryptography. It has a very specific meaning to specialists but is widely used by non-specialists in ways that invite confusion. True randomness is very difficult for humans to generate. Computers also cannot really do it, although they are great at creating as many pseudo-random letters or numbers as you like.

2 Simple Substitution Ciphers

Substitution is one of the easiest ways of “hiding” text – you simply replace one letter with another letter or perhaps a number or symbol. Sounds simple, but the catch is in how you replace each letter. It has to be done in a way that lets both the sender and the receiver encipher / decipher accurately (quickly would be nice to, but accuracy is more important). In other words, both sides must know the algorithm (a fancy way of saying “process”) for replacing each letter. As a practical matter, encoding / decoding algorithms that involve remembering huge charts or going through 20 separate steps are no good – people just won't do it. So the method has to be easy to use.

There are a huge number of potential substitution ciphers. Using the letters of a different alphabet to encode is one way. The Japanese language does this with something called “romaji” – the Japanese language written in the Latin (Roman, hence “roma”) alphabet. Romaji is theoretically a part of the Japanese language (something bolted on to the language some might say) but for many Japanese people, romaji is a cipher they have to deal with on a daily basis. In this chapter, we will use English for both the plain text and the cipher alphabet.

Another method of substitution is to convert the letters (of whatever alphabet) into numbers. This in turn opens up a host of opportunities for further encipherment, because you can do math on numbers much more easily than on letters. Historically, this was a huge step forward for ciphers and its importance is not limited to substitution ciphers. We will look at these ciphers in a later chapter.

One of the earliest recorded ciphers is the one named for Julius Caesar – the Caesar Cipher. It is very easy to use, but is also very easy to break.

The Caesar Cipher

The Caesar Cipher is a very simple mono-alphabetic substitution. Mono-alphabet means what it sounds like, namely that there is only one alphabet used for enciphering the plain text. Every plain text letter has one and only one corresponding cipher text letter.

In the case of the Caesar cipher, the alphabet is simply shifted three spaces and each letter of the plain text is replaced by the new letter. So a becomes D and p becomes S.

Plain text	a	b	c	d	e	f	g	h	i	j	k	l	m
Cipher text	D	E	F	G	H	I	J	K	L	M	N	O	P

Plain text	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher text	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

The only things to remember with the Caesar cipher are the number of letters to rotate the alphabet and the direction of rotation. In this case, the plaintext is rotated three letters clockwise. The number must be agreed upon

by both parties in advance, but can be any number from 1 to 25 (26 would result in the plaintext and ciphertext being the same). Note that the amount of rotation, whether three (as in the example above), 12, 17, or 25, has no affect on the difficulty of unauthorized decryption.

Since there are no charts or other difficult processes to remember, this system is easy to use. In a time of widespread illiteracy, it might have even been secure. Today it is trivial and not worth the time it takes to encode and decode. It won't stop anyone who really wants to read you're mail. It will stop casual observers from reading it if they aren't willing or able to expend a small amount of effort.

There are many ways to make a substitution cipher like the Caesar stronger (ie harder for the bad guy to break) while keeping the basic usage the same. One way is the use of keywords to scramble the alphabet before substituting.

Key Words

One problem with the Caesar cipher is that the letters of the alphabet are all still in order – `a' comes right before `b' and `o' comes right after `n'. This is a big weakness because it gives the bad guys some information. For example, this Caeser cipher text \tpt{wklv} tells us a lot about the plaintext. Because `w' and `v' are next to one another in the alphabet, we know that the plaintext letters in those positions must also be next to each other. The same is true for `k' and `l'. Further, we know that the the plaintext of the cipher `w' and `k' must be the same distance apart, specifically twelve places. This information won't tell us what word is encoded by wklv, but it does tell us what words aren't, as well as what words it *might* be. It cannot be the name `Mark', for example, since the letters `m' `a' `r' and `k' do not have the characteristics that the ciphertext has.

A simple way to avoid giving away so much information, is to use a keyword. Suppose we use the word, `saturday'. Write out the alphabet normally and then below it write the keyword, each letter only once, (drop the second `a' in `saturday'). After the keyword, continue the alphabet, skipping any letter that is in the keyword. It would look like this:

Plain text	a	b	c	d	e	f	g	h	i	j	k	l	m
Cipher text	S	A	T	U	R	D	Y	B	C	E	F	G	H

Plain text	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher text	I	J	K	L	M	N	O	P	Q	V	W	X	Z

Assuming that the keyword is easy to remember, this system is a little more secure than a regular Caesar cipher because it gives the enemy less information, but is not significantly harder to use. Unfortunately, anything en-

ciphered with a Caeser plus a keyword is still rediculously insecure, even by the standards of traditional ciphers.

Of course, the longer the keyword, the more mixed the resulting alphabet is, assuming that it has many different letters, and the more mixed it is, the more effort is required to break the code. The word `success' is no better a keyword than `bear', since success has only four different letters. Further, unless you use a very long keyword or a key phrase, the remaining letters of the alphabet do not get mixed–everything after the `y' from saturday in the example above is in alphabetical order, with some missing letters.

Another weakness of the keyword is that it is probably from the same language as the plaintext. This means that the letters of the keyword will themselves contain some information. It would be very unusual for an English keyword to have the sequence `vzx', however `th', `ng' and `gh' would not be uncommon combinations in an English keyword. More importantly, it limits the number of possible cipher alphabets.

One step (slighly) more complex than a Caesar with a keyword is a substitution cipher that uses a keyword to generate a new alphabet -- a new order that is, the letters of the alphabet remain the same.

Mixed Alphabets

Instead of using the keyword at the head of the alphabet (as with saturday, above), make a grid with the keyword as the first row. Then we use columnar transposition[1] to create a new order for the alphabet. For example, let's use the keyword `loquacious':

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

After filling in the table like above, we read off the letters in columns, top to bottom, left to right (although you could do it any direction you wanted). This gives us the mixed cipher alphabet below.

Plain text	a	b	c	d	e	f	g	h	i	j	k	l	m
Cipher text	L	B	M	Y	O	D	N	Z	Q	E	P	U	F

Plain text	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher text	R	A	G	T	C	H	V	I	J	W	S	K	X

This alphabet is still not random but it is much better than any of the previous ones we've looked at. A message `send help now` becomes:

1 For more on columnar transposition, see chapter 3 on page ??.

Plain text	s	e	n	d	h	e	l	p	n	o	w
Cipher text	H	O	R	Y	Z	O	U	I	R	A	W

There are two important things to notice in the above ciphertext. First, the fact that plaintext \tpt{w} became ciphertext \tct{w}. A letter can become itself in the ciphertext. Why not? If this isn't a possibility, you are limiting the number of possible ciphertext alphabets and in a long message, it could be noticed. During World War II, an Italian sent out a fake message composed of nothing but plaintext \tpt{l} repeated over and over. The Italians were not using simple substitution and the ciphertext alphabet changed after each letter. However, in the Italian code \tpt{l} could never become \tct{l} and thus \tct{l} was not used in that broadcast. An observant Allied cryptographer noticed this, guessed the cause, and used that information as a wedge to pry open the entire cipher.\footnote{Find the citation for this–think it is in Kahn.} Lesson: there is no reason not to let the cipher letter and plaintext letter be the same, if that is what happens with the algorythm you are using.

More important than \tpt{w} becoming \tct{w} is the fact that plaintext \tpt{e} became ciphertext \tct{o} twice. Similarly, plaintext \tpt{n} also became ciphertext \tct{r} twice. This is the biggest weakness of simple substitution–the alphabet is always the same, so letter frequencies do not change. Letter frequencies make every simple substitution cipher insecure, regardless of keywords or mixed alphabets.

Letter Frequencies

The problem with human languages is the fact that they discriminate–not every possible combination of sounds or letters is used by the language. Of course, which combinations are used or allowed is different from language to language. Japanese, for example does not have the sounds which corresponds to the English letters `l' or `v'. Sounds that are one letter in some languages are two or even three letters in other languages. And every language has combinations of sounds / letters that are just not allowed. English speakers, please try to pronounce the word `bzxdfpq'. You can't, because in English that is not an allowable combination of letters.

Languages also prefer some letters more than others. Some combinations occur more than others. In English, `qui' is acceptable, but it does not occur as often as `the', but occurs much more often than `gry'. Finally, as most English speakers are aware, `e' occurs far more often than any other letter. Although the commom and uncommon letters will vary with the language, the fact that some letters occur more than others will not.

The troublesome fact is that the frequency of letters in any given language is fairly stable regardless of the context. With a long enough passage, the letter frequencies of a legal document and a transcript of a conversation between two people will show little variation – they will be almost the same.

This is bad for the person who wants to use a substition cipher. Simple substition – where there is only one plaintext alphabet and one cipher alphabet – is totally insecure because the letter frequencies will tell the decipherer

which letters are which. For example, Figure \ref{fig:thoreau_1} shows the statistics for the Project Gutenburg version of Henry David Thoreau's *On the Duty of Civil Disobedience* (minus the title and the Project Gutenburg small print). The total number of letters is 40,071.

\caption Letter Frequencies from *On the Duty of Civil Disobedience*
\label{fig:thoreau_1}

Letter	Number	%	Letter	Number	%
a	3127	7.8037	n	2921	7.2896
b	592	1.4774	o	3200	7.9858
c	1068	2.6653	p	675	1.6845
d	1439	3.5911	q	45	0.1123
e	4962	12.3830	r	2237	5.5826
f	860	2.1462	s	2559	6.3862
g	643	1.6047	t	4219	10.5288
h	2410	6.0143	u	1057	2.6378
i	2946	7.3520	v	514	1.2827
j	74	0.1847	w	858	2.1412
k	182	0.4542	x	73	0.1822
l	1480	3.6934	y	845	2.1088
m	1069	2.6678	z	16	0.0399

In just over 40,000 letters, there are almost 5,000 `e's but only 16 `z's. Figure \ref{fig:thoreau_2} shows the letter frequencies for the same text, after putting it throught a simple substitution, rotating the letters by five. Of course the numbers are the same. The substitution has not hidden them at all, just moved them five places.

```
\begin{figure}
\caption{Letter Frequencies from \textit{On the Duty of Civil
   Disobedience} (Rotated Five Places)}
\label{fig:thoreau_2}
\begin{center}
\begin{tabular} {|c|r|r||c|r|r|}
\hline
letter&number&percent of total&letter&number&percent of total\\
\hline
a&514&1.28272&n&2946&7.35195\\
b&858&2.1412&o&74&0.184672\\
c&73&0.182177&p&182&0.454194\\
d&845&2.10876&q&1480&3.69344\\
e&16&0.0399291&r&1069&2.66776\\
f&3127&7.80365&s&2921&7.28956\\
```

```
g&592&1.47738&t&3200&7.98583\\
h&1068&2.66527&u&675&1.68451\\
i&1439&3.59113&v&45&0.112301\\
j&4962&12.383&w&2237&5.58259\\
k&860&2.14619&x&2559&6.38616\\
l&643&1.60465&y&4219&10.5288\\
m&2410&6.01432&z&1057&2.63782\\
\hline
\end{tabular}
\end{center}
\end{figure}
```

The stats are the same, just rotated five letters. So the stats for `a' in the plaintext are the same as the stats for the ciphertext letter `f'. In this case, there are almost 5,000 `j's in the ciphertext–far more than any other letter–so it is a safe bet that `j' is equal to `e'. (Of course, the only way to be sure is to plug `e' in and see what results.)

Solving Simple Substitution

Mixed alphabets help to avoid some of the problems of letter frequency. With a plain vanilla Caesar-type cipher, once the bad guy gets a good idea of the most common letters, he has practically read the message. If I can guess at `e', not only can I guess at `d' and `f', I can also get some confirmation if the frequencies of other common letters fall into place around the `e'. This is because simple substitution does nothing about the distribution of the letter frequencies. In the example above, `e' and `t' each account for over 10% of the letters in the plaintext. They are fifteen letters apart. In the ciphertext, `j' occurs over 10% of the time and fifteen letters later, `y' also occurs over 10% of the time.

While a mixed alphabet does nothing to change the letter frequencies, it does change the distribution of the frequencies. This can make decryption a bit more difficult for the bad guys – it denies them a little bit of information.

Of course, the bad guy usually doesn't have access to the plaintext and the ciphertext. Unfortunately, it doesn't matter. The statistics of the language will be the same. In any sufficiently long message, the frequencies of the letters will not differ much from the expected frequencies for that language. So with English, there will almost always be a lot more `e s t r n i o` than any other letters. If the bad guys have the ciphertext, they can get the plaintext.

Step one: count the letters and determine their frequencies.

Step two: plug in possible matches and see how things look. Any impossible combination of letters means that one of the letters probably isn't correct. Combinations that are okay should suggest new possibilities. The process continues until the message is decrypted.

That's it. It doesn't matter if the cipher used a mixed alphabet or not. A well mixed alphabet just makes it the cipher less easy to break. Notice also

that the bad guy doesn't need to find the keyword, if one was used to mix the alphabet. This is important enough to repeat: *The message can be read without knowing the keyword*.

There are still a few things that you can do to make life more difficult for the decrypter. Don't use spaces or punctuation. Intentionally mis-spell words. Leave out the second letter if there are two in a word (balloon becomes balon, for example). Use `nulls'–letters that have no meaning and are included just to confuse the decrypter. Good candidates are `q' since it almost never occurs outside of `qu' and `i' or `j' if you let one stand for both and use the other as a null. Using nulls as punctuation markers defeats the purpose of using nulls in the first place.

All of these have been used but even with them, any substitution cipher is vulnerable. All they can do is slow the attacker down a little. Which is why long ago people started using substitution ciphers that weren't so simple.

3 Not So Simple Substitution

As we have seen, substitution ciphers involve replacing one letter of plaintext with a letter of ciphertext. In simple substitution ciphers, there is only one replacement ciphertext letter for each letter of the plaintext alphabet. Because simple substitution ciphers do nothing to hide the letter frequencies – at best they can mix the distribution of the frequencies – they are totally insecure.

What we need to do is find a way to hide those letter frequencies. One way is to add some extra letters to the cipher alphabet and use them to give common letters more than one ciphertext letter. If an \tpt{e} becomes an \tct{m} in one place and an \tct{s} somewhere else, then the high frequency of the \tpt{e} in the plaintext will not be seen in the ciphertext – indeed, it will be split in half. As an example, below is the Caesar cipher with a small twist: I've added the numbers 0-9 as alternate replacements for common letters.

```
\begin{tabular}{l|c|c|c|c|c|c|c|c}
Plaintext:&\tpt{a}&\tpt{b}&\tpt{c}&\tpt{d}&\tpt{e}&\tpt{f}&\tpt{g}&\tpt
{h}&%
\tpt{i}\\
\hline
Ciphertext:&\tct{d 0}&\tct{e}&\tct{f}&\tct{g}&\tct{h 9 1}&\tct{i}&\tct{j}
&\tct{k}&%
\tct{l 8}\\
\end{tabular}

\vspace{1em}

\begin{tabular}{l|c|c|c|c|c|c|c|c}
Plaintext:&\tpt{j}&\tpt{k}&\tpt{l}&\tpt{m}&\tpt{n}&\tpt{o}&\tpt{p}&\tpt
{q}&\tpt{r}\\
\hline
Ciphertext:&\tct{m}&\tct{n}&\tct{o}&\tct{p}&\tct{q 2}&\tct{r7}&\tct{s}
&\tct{t}%
&\tct{u 3}\\
\end{tabular}

\vspace{1em}

\begin{tabular}{l|c|c|c|c|c|c|c|c}
Plaintext:&\tpt{s}&\tpt{t}&\tpt{u}&\tpt{v}&\tpt{w}&\tpt{x}&\tpt{y}&\tpt
{z}\\
\hline
Ciphertext:&\tct{v 6 4}&\tct{w 5}&\tct{x}&\tct{y}&\tct{z}&\tct{a}&\tct
{b}&\tct{c}\\
\end{tabular}
```

Using this table, we can encode \tpt{send help now} as:

```
\begin{center}
\begin{tabular}{lccccccccccc}
\tpt{s}&\tpt{e}&\tpt{n}&\tpt{d}&\tpt{h}&\tpt{e}&\tpt{l}&\tpt{p}&%
\tpt{n}&\tpt{o}&\tpt{w}\\
\hline
\tct{v}&\tct{h}&\tct{q}&\tct{g}&\tct{k}&\tct{9}&\tct{o}&\tct{s}&%
\tct{2}&\tct{r}&\tct{z}\\
\end{tabular}
\end{center}
```

Notice that no letter appears more than once.

In the above example, the individual letter frequencies are much reduced – there is less variation between frequencies. In plaintext the difference between the frequency of \tpt{e} and \tpt{z} is large and this makes both letters easy to spot. But with a well designed substition the frequencies of all the letters are leveled and ideally they should all approach about 4% (roughly four of each of 26 letters for every 100 letters) or less if you were using more letters (as in the example, where we our cipher alphabet has 36 letters to encipher 26 plaintext letters).

Still Not Good Enough

However, even assuming your substitution alphabet managed to totally even out the frequencies, it would still be very weak. Why? It is weak because in the real world there are limits on how complex a system can be. After all, humans must use the system and enciphering and deciphering must be managable with limited training and perhaps limited materials. A spy in hostile territory cannot carry around a huge chart with all sorts of substitutions or code phrases. Front line military units cannot afford to spend 30 minutes deciphering a message that says ``attack now''.

So reality will constrain the system to some degree. Which is where bigram and trigram frequencies will weaken even a substitution system that neutralizes individual letter frequencies. Given enough ciphertext, the enemy will be able to read your messages.

Let's look at an example.

4 Transposition Ciphers

Transposition ciphers involve changing the place of the plaintext letter in the message. The scramble-grams that many newspapers carry are simple examples of transposition ciphers. For example, \tpt{help} might become \tct{eplh}. The letters are all the same, just their position has changed.

As with substitution ciphers, the difficulty is not enciphering the plaintext. Rather, it is enciphering the message in such a way that your friends can easily decipher it but your enemies cannot. Just randomly mixing up the let-

ters won't do because how will your recipient know that they have unscrammbled the letters into the correct message?

Like every other type of cipher, the transposition cipher depends on an algorithm – which the sender and recipient must agree on beforehand. One very simple algorithm is writing the message backwards – so \tpt{help} $\rightarrow$ \tct{pleh} but that is hardly secure. A common way of mixing the letters is the columnar transposition.

Columnar Transposition

With columnar transposition, you write the message into a rectangle by rows and then read it off by columns. Hence the name. There are of course plenty of ways to do this – exactly how is the `key' that lets your friends read the message put makes it hard for others to do so.

Say we want to send the following message:

```
\begin{tt}
Negotiations are proceeding as per plan. Expect to \\
finalize everything next week. Email any last minute \\
changes to my private address.
\end{tt}
```

First, we drop capitalization, punctuation, and spaces. Although at this stage it isn't important, lets break the resulting message into five letter groups, just to make it a bit easier on our eyes. This gives us:

```
\begin{tt}
negot iatio nsare proce eding asper plane xpect\\
tofin alize every thing nextw eekem ailan ylast\\
minut echan gesto mypri vatea ddres s
\end{tt}
```

Now, we must have a `key' that we know and that our recipient knows. Pretend we decided previously on an eleven column table. Writting the message into such a table, we get this:

```
\begin{center}

\textbf{Step One of Columnar Transposition}

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
1&2&3&4&5&6&7&8&9&10&11\\
\hline
\tpt{n}&\tpt{e}&\tpt{g}&\tpt{o}&\tpt{t}%
&\tpt{i}&\tpt{a}&\tpt{t}&\tpt{i}&\tpt{o}&\tpt{n}\\
```

```
\hline
\tpt{s}&\tpt{a}&\tpt{r}&\tpt{e}&\tpt{p}%
&\tpt{r}&\tpt{o}&\tpt{c}&\tpt{e}&\tpt{e}&\tpt{d}\\
\hline
\tpt{i}&\tpt{n}&\tpt{g}&\tpt{a}&\tpt{s}&%
\tpt{p}&\tpt{e}&\tpt{r}&\tpt{p}&\tpt{l}&\tpt{a}\\
\hline
\tpt{n}&\tpt{e}&\tpt{x}&\tpt{p}&\tpt{e}%
&\tpt{c}&\tpt{t}&\tpt{t}&\tpt{o}&\tpt{f}&\tpt{i}\\
\hline
\tpt{n}&\tpt{a}&\tpt{l}&\tpt{i}&\tpt{z}%
&\tpt{z}&\tpt{e}&\tpt{e}&\tpt{v}&\tpt{e}&\tpt{r}\\
\hline
\tpt{y}&\tpt{t}&\tpt{h}&\tpt{i}&\tpt{n}%
&\tpt{g}&\tpt{n}&\tpt{e}&\tpt{x}&\tpt{t}&\tpt{w}\\
\hline
\tpt{e}&\tpt{e}&\tpt{k}&\tpt{e}&\tpt{m}%
&\tpt{a}&\tpt{i}&\tpt{l}&\tpt{a}&\tpt{n}&\tpt{y}\\
\hline
\tpt{l}&\tpt{a}&\tpt{s}&\tpt{t}&\tpt{m}%
&\tpt{i}&\tpt{n}&\tpt{u}&\tpt{t}&\tpt{e}&\tpt{c}\\
\hline
\tpt{h}&\tpt{a}&\tpt{n}&\tpt{g}&\tpt{e}%
&\tpt{s}&\tpt{t}&\tpt{o}&\tpt{m}&\tpt{y}&\tpt{p}\\
\hline
\tpt{r}&\tpt{i}&\tpt{v}&\tpt{a}&\tpt{t}%
&\tpt{e}&\tpt{a}&\tpt{d}&\tpt{d}&\tpt{r}&\tpt{e}\\
\hline
\tpt{s}&\tpt{s}&\tpt{}&\tpt{}&\tpt{}%
&\tpt{}&\tpt{}&\tpt{}&\tpt{}&\tpt{}&\tpt{}\\
\hline
\end{tabular}

\end{center}
```

Now we read off the letters by the column. The simplest way is to just start with at the top of column one, write down the letters from that column and then start again at the top of column two, continuing through all the columns. This would give of cipher text looking like this (broken into five letter groups):

```
\tct{nsinn} \tct{yelhr} \tct{seane} \tct{ateaa} \tct{isgrg}
\tct{xlhks} \tct{nvoea} \tct{piiet} \tct{gatps} \tct{eznmm}
\tct{etirp} \tct{czgai} \tct{seaoe} \tct{tenin} \tct{tatcr}
\tct{teelu} \tct{odiep} \tct{ovxat} \tct{mdoel} \tct{fetne}
\tct{yrnda} \tct{irwyc} \tct{pe}
```

The recipient must do a little math before decoding, but only a little. Basically decoding just involves writing the message into columns and then reading the message from the rows – the reverse, naturally enough of how we encoded it. However, the key is the number of columns, and says nothing about the number of rows. Look at the table we wrote the original message into. The table ended up being a square, 11 columns and eleven rows, but this is just a coincidence – a different message could have had any number of rows, depending on the length of the message. Further, only columns 1 and 2 have a letter in the final row (both `s' in this case).

In order to know how many rows to use, the decoder divides the length of the message by the number of columns (the key, which they need to know anyway). In our example, the message has 112 letters and there are 11 columns. Since 112 divided by 11 is 10 with a remainder of 2, the recipient knows to make a table with 11 columns and 11 rows but to only put letters in the first two columns of the eleventh row, leaving the rest blank. Of course, the enemy doesn't know the key and so doesn't know what to divide by to find the number of rows.

What It Does and Doesn't Do

Any sort of transposition cipher breaks the digrams and trigram frequencies of the plain text. So repeated `th', `ing', `es', etc. which cause substitution ciphers such grief are not a big problem for transposition ciphers.

However, they do nothing to hide the real letters. All the letter frequencies are the same as for plain text. So the cipher text and the plain text from above will show exactly the same letter frequencies and for a long enough message those frequencies will be very close to normal English letter frequencies. So if you see a cipher where cipher text frequencies for \tct{e} and \tct{t} are high and \tct{z} is very low, but the message looks like gibberish and there are no good frequency matches for digrams or trigrams, there is a good chance that you are looking at a transposition cipher.