2 Historical Manual Methods

In this section you will learn about some of the historical ways that humans used to protect text and messages. You'll start out by learning about methods that were used to hide messages without scrambling the text, and then learn about the earliest methods for scrambling messages. All the methods are amazingly creative and the stories behind their use are very interesting. There are stories of how hiding messages or breaking ciphers helped win battles or topple empires, or how putting your faith in an "unbreakable" system might cost you your head.

Besides the stories you will be introduced to some actual historical methods that were developed for hiding text. These methods are all "manual" methods in the sense that you can sit down with a pencil and paper and use them to encrypt or decrypt messages, no computer is required. While these older methods aren't used for protecting data anymore, at least in their manual forms, it's still important to learn how they work as they are the foundation of current methods.

The main goal of this section is for you to gain a good working knowledge of substitution ciphers, as they are the first step in the evolution of modern methods.

The specific things you should be able to do at the end of this section are:

1. Define the terms steganography, cryptography, and cryptanalysis; and describe how they are different and how they are related.
2. Define the terms codes and ciphers, and describe how they are different.
3. List and describe the general methods used by substitution and transposition ciphers.
4. Given a specific substitution or transposition cipher, use it to encrypt and decrypt text.
5. Explain the principles behind frequency analysis.
6. Use frequency analysis to decipher cipher text characters and messages.
7. Use frequency analysis to determine whether a message has been encrypted using a substitution cipher or a transposition cipher.
8. Describe methods used to thwart frequency analysis.

Required Reading

In addition to reading this document you should read or view the following:

1. Singh – Chapter 1 and Chapter 5 pages 191- 201
2. Singh – Chapter 2 pages 52-54

3. Go to the Crypto Corner Website - https://crypto.interactive-maths.com/
   Read all of the pages and do all of the activities under the Monoalphabetic Substitution Ciphers and the Simple Transposition Ciphers sections.

4. Go to the Khan Academy Journey Into Cryptography Website – https://www.khanacademy.org/computing/computer-science/cryptography
   View the following videos under the Ancient cryptography section:
   - What is cryptography?
   - The Caesar cipher
   - Caesar Cipher Exploration
   - Frequency Fingerprint Exploration

5. http://practicalcryptography.com/ciphers/homophonic-substitution-cipher/

6. https://www.simonsingh.net/The_Black_Chamber/homophonic_cipher.html

Optional Reading and Viewing
There are hundreds of web sites and web pages with information on cryptography. Here are a few of the more interesting and more useful sites that I've found over the years.

https://www.wired.com/2012/11/ff-the-manuscript/ - Wired article "THEY CRACKED THIS 250-YEAR-OLD CODE, AND FOUND A SECRET SOCIETY INSIDE"

https://www.youtube.com/watch?v=Dq4DDybD1ww - The Adventures of Sherlock Holmes S01E02 The Dancing Men (Note – this isn't the great series with Benedict Cumberbatch and Bilbo Baggins (Martin Freeman), or even the semi-great movies with Robert Downey Junior.)

http://www.historyofinformation.com/detail.php?id=4168 - A description of the scytale and how it functions.

https://www.dcode.fr - This site has a tool for simulating several ciphers, and performing frequency analysis

https://vimeo.com/188198031 TRUE WHISPERS: The Story of the Navajo Code Talkers

https://crypto.interactive-maths.com/pigpen-cipher.html

https://inventwithpython.com/cipherwheel/ - Try spinning the cipher wheel yourself

https://access.redhat.com/blogs/766093/posts/1976023 - A Brief History of Cryptography

http://rumkin.com/tools/cipher/ - tools for encrypting and decrypting caesar, rot, vigenere, etc.

https://www.guballa.de/substitution-solver - tool for solving substitution ciphers

https://quipqiup.com - tool that will solve substitution cipher, and show various possible solutions.

http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/hints.html - hints for solving cryptograms and substitution ciphers

https://www3.nd.edu/~busiforc/handouts/cryptography/cryptography%20hints.html - hints for solving cryptograms and substitution ciphers

Section Content

The First Recorded Cryptographic Methods

The first two cryptographic methods you're going to learn about are significant because they're the first methods in any written historical record. These methods are the scytale and steganography. It's possible that there were other ways of hiding message before this, as I'm sure you can think of ways you could simply hide a message, like sticking it in your shoe or in your hat. But these methods are a little cleverer than that, and they are the earliest found in recorded history. Plus, they're both good examples of human ingenuity.

Scytale

The scytale method for hiding data is a clever scheme that most books mention because it's one of the earliest recorded methods of hiding information. There's a little academic dispute about the first mention, it may have been as early as 7th century BC if you include an indirect mention. But it was definitely written about in the 3rd century BC.

It involved wrapping a strip of leather around a rod and then writing the message down the length of the rod. The strip is then unwrapped for transport. Anyone who sees the strip of leather only sees what looks like a random string of characters. When the recipient receives the message, they wrap it around another rod of the same diameter, and the message reappears.

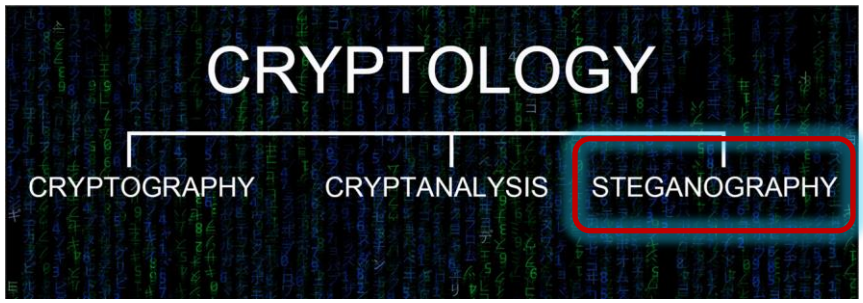
The scytale method is actually a special form of a transposition cipher (which you'll learn about later).

It's not crucial that you know the details of how a scytale works, you're just learning about it because of our society's obsession with firsts, and it's one of the first recorded methods for scrambling data. If you do want to know more you might check out the following resources:

Steganography

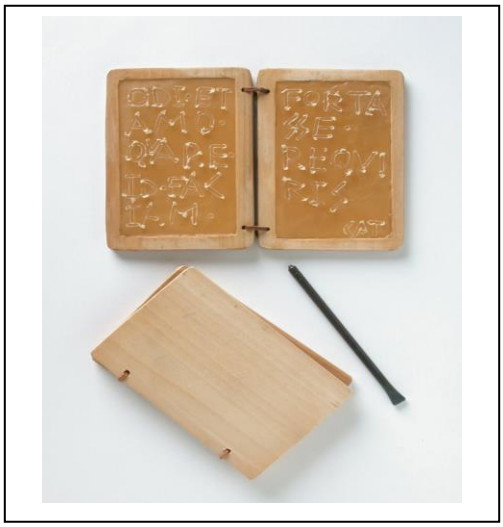
The next historical methods of mention were all forms of steganography.

Steganography is one of the main subfields of the parent science of Cryptology, and it includes several different methods for hiding text. The key feature that distinguishes steganography from cryptography is that in steganography the text is hidden but not scrambled, as opposed to cryptography where the text is scrambled but not necessarily hidden. Since with steganography the text was not scrambled, it meant any intercepted message could be easily read. The message sender had to assume that the person delivering the message would be searched, which is why they had to find better ways of hiding messages.



Here are a couple of interesting stories from history involving steganography.

One of the first written historical references to the use of steganography was made by Herodotus in the book *The Histories* when he described the war in 500 B.C. between the Persians and the Greeks. You may be familiar with the story of these battles from the movies *The 300* and *The 300: Rise of an Empire*. While good portions of these movies were fictionalized to make the story more entertaining, the main stories about the battles are true. The Persians led by Xerxes were an immense Empire that wanted to conquer the Greeks. The Persians did have a giant fleet led by Artemisia that they planned to use to invade Greece.



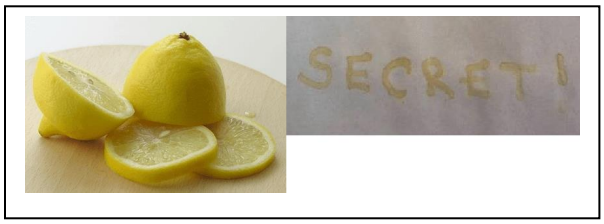
The movie skips over how the Greeks learned of the invasion, but Herodotus[1] describes what is considered to be the first historical reference to steganography. A Spartan exile named Demaratus was living in Persia and learned of the invasion plans. He wanted to warn the Greeks, but also worried about the dangers being discovered. At the time many messages were written on wax covered wood tablets. A stylus was used to inscribe the message in the wax. And the tablet could be reused by simply melting the wax. Demaratus came up with the plan to write his warning on the wood portion of the tablets, and then cover over the writing with some wax so the tablets looked blank. He was able to send the tablets back to Greece, sneaking them right past the guards who were watching the roads. The guards only saw the blank wax when they inspected the tablets and allowed them to pass. When the tablets reached Greece luckily a woman named Gorgo, the wife of Leonidas, was smart enough to melt the wax and read the message. Herodotus doesn't explain how Gorgo knew about the hidden message, only that she knew they would find a message if they scraped off the wax. Finding the message allowed the Greeks to prepare for the Invasion. I won't tell you how it all ends so it won't spoil the ending if you decide to read the story or watch the movie.

Herodotus also writes another story about a different form of stenography where Histiaeus, another Greek living in Persia, shaves the head of one of his slaves and tattoos a message on

[1] http://mcadams.posc.mu.edu/txt/ah/Herodotus/Herodotus7.html

the slave's bald head. He then waits until the slave's hair grows enough to hide the message and then sends the slave to deliver the message along with instructions on how to read it. The entire story is a huge soap opera, full of betrayal and revenge. But the important thing to note is that these were some of the earliest recorded uses some rather clever methods of hiding messages and most people assume that the stories are true. As a parenthetical note I say that most people *assume* they're true because if you actually read Herodotus' *The Histories* you'll find some stories that are a little out there. For example, *The Histories* contain a tale about giant furry ants that mine for gold and were fast enough to outrun an adult camel[2].

Here are a couple of other methods of steganography that you may encounter today.



The first is using "invisible ink". This can be as simple as the kid's project where you write the message in lemon juice. When the lemon juice dries out it's invisible. But if the paper is heated, the writing shows up again. Throughout history a variety of fluids were used to create the invisible ink effect[3]. There's also a modern twist to invisible ink, where you write with an ink that, after it dries, only shows up under uv light.

These ancient methods of steganography are fun to read about, but as you can see they require quite a bit of time to deliver the message, months in both of the historical cases, but even days if you use invisible ink and have to mail the message. Today there's an electronic form of steganography that's makes delivery times quicker. This method involves taking the bits representing the data from the message file and mixing them in with the bits of another file. This is typically done by mixing the data bits into an image file. Here's a brief explanation of how this works and why it works.

First, let's look at how image data is stored. And just to be technically correct, we're talking about images like photos, not images that are created from vectors. The data for a photo is made up of a set of pixels, each of which represents a small point of color at a specific location in the image. The data for each pixel consists of a set of numbers representing the amount of red, the amount of green, and the amount of blue. The size of each red, green, and blue number can vary, but most jpegs use numbers that range between 0 and 255 for each color. Think of this as having 3 different lights, a red light, a green light, and a blue light. You have controls for each light and can vary the brightness from 0 where the light is completely off to 255 where the light is as bright as it can be. The resulting color will be combination of all three lights.

The colors for the RGB values at each pixel are actually 8-bit numbers, that range from 00000000_2 to 11111111_2. There two important things to note about this system. The first is that when these numbers are combined, they result in over ~16.7 million different possible color combinations, which is a lot of different colors. The second thing to note is that changing the smallest value in any of the RGB numbers results in colors that are very close together, and very difficult to tell apart. For example, the red value $0111111\mathbf{0}_2$ and the red $0111111\mathbf{1}_2$ are obviously different numbers, but the resulting colors look almost identical.

---

[2] https://theglyptodon.wordpress.com/2011/05/20/herodotus-vs-the-ants/

[3] https://www.artofmanliness.com/articles/man-knowledge-the-history-of-invisible-ink/

Red 0111111**0**$_2$

Red 0111111**1**$_2$

The next piece of the steganography system is the message you want to hide. The important concept here is that on computers all data is stored as binary numbers. It doesn't matter if it's text data, images, sounds, movies, etc. all data is stored as 0's and 1's. To create a simple example, let's assume we want to hide the text message **At 2:41 PM**. The first character in this message is the letter **A**. The ASCII value for **A** is 65_{10} or 01000001_2.

The actual hiding is done by combining the bits from the data file with the RGB data from the image file. The combining works like this:

- Take the first bit from the data. In the case of **A** or **0**1000001_2 the first bit is a 0.
- Insert this into the least significant bit of the Red value for the first pixel in the image. For example, if the Red value is 0111111**1**$_2$ change it to 0111111**0**$_2$. But if the Red value is the 0111111**0**$_2$ inserting the 0 from the **A** data will leave the Red value unchanged.
- Take the second bit from the data. In the case of **A** or 0**1**000001_2 the second bit is a 1.
- Insert this into the least significant bit of the Green value for the first pixel in the image. For example, if the Green value is 0101001**0**$_2$ change it to 0101001**1**$_2$. But if the Green value is the 0101001**1**$_2$ inserting the 1 from the **A** data will leave the Green value unchanged.
- Take the third bit from the data. In the case of **A** or 01**0**00001_2 the third bit is a 0.
- Insert this into the least significant bit of the Blue value for the first pixel in the image. For example, if the Blue value is 1001100**1**$_2$ change it to 1001100**0**$_2$. But if the Blue value is the 1001100**10**$_2$ inserting the 0 from the **A** data will leave the Blue value unchanged.
- Repeat this process by combining the next 3 bits from **A** with the Red, Green and Blue pixel values for the second pixel.
- Repeat the entire process using subsequent characters from the text, with subsequent Red Green and Blue pixel values from the image.

When the process is completed the data bits from the original text message will be embedded in the image. The recipient can rebuild the original message by reversing the process. They will run a program that grabs the least significant bit from each Red, Green and Blue value for each pixel, and uses them to reconstruct the original ASCII character values.

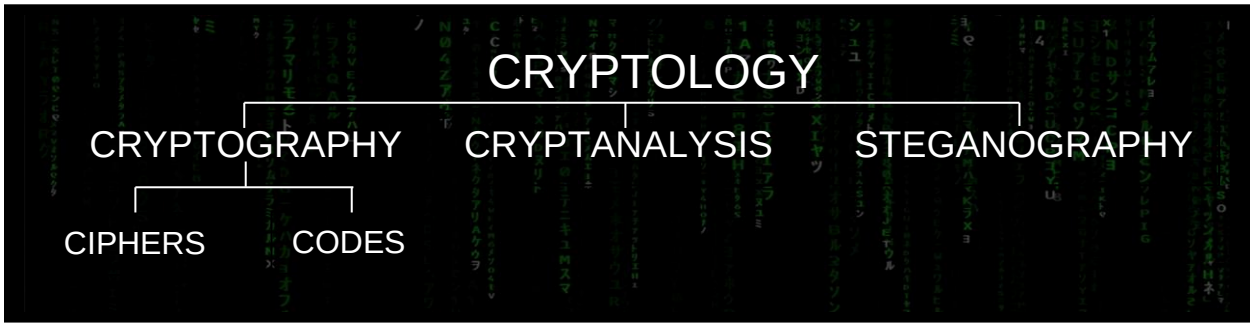
The sneaky thing about this is that no one can tell the image has been altered. This is because changing each color number by 1 results in changes so small that they will be invisible to visual inspection. The only way to tell the image has been changed is if you have a copy of the original image. With a copy of the original image, it's possible to compare the least significant bits for each pixel, and if they're changed it's a sign that other data has been inserted. But without a copy of the original image there's no way to tell that the pixel data has been altered.

While this method of modern steganography is interesting, it's not crucial to the study of modern cryptography as it's a completely separate field.

Cryptography

The other main branch of cryptology is cryptography. The first cryptographic methods were invented because hiding messages with steganography was risky. Since the message was in plain text, if the message was found the secret was lost. This led people to look for different ways to keep their communications private. With cryptography it was assumed that people besides the recipient are going to see the message, so they need to be scrambled to make them impossible or difficult to read. And cryptography is the branch of cryptology that's behind the modern methods, so it's where we'll concentrate most of our time and effort.

In cryptography there are two main ways to scramble the text and make it difficult to read or understand. The first method is to use a ***code*** which replaces words and phrases with other words and phrases. The second method is to use a ***cipher*** which replaces each character in a message with another character. You will first learn more details about codes and ciphers, and then do a deeper dive into some specific ciphers.

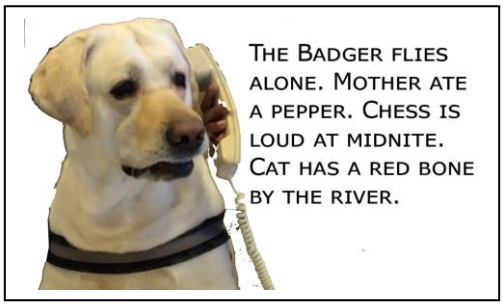


Codes

Using a code means changing entire words and phrases for other words and phrases. The process of changing plain text using a code is called encoding, and the process of changing coded text back to plain text is called decoding. In this section you'll learn about two different methods that have been and can be used to encode messages. The first is to change the plain text words for words in the *same* language, and the second is to translate the plain text words to a *different* language.

Spy Talk

The first method of using codes is to change words and phrases to different words or phrases in the same language. This creates messages which sound like typical movie "spy talk". For example, the sender and the recipient may agree that word **king** will be replaced by the word **sword** and the word **castle** will be replaced by the word **pickle**. The phrase **the sword is in the pickle** would be decoded as **the king is in the castle**.



For this to work the sender and the recipient obviously must use the same translation table, which presents a few problems. The first is coming up with the code in the first place. It might seem easy to do, and it probably is relatively easy if you only have to create one code, one time. But if you think about how many different words and phrases we use it could be difficult to repeatedly come up with different translation tables that cover every possible word. The research that I've read[4] says that most adults know 20,000 – 35,000 different words, but typically only use about 1000 words for ~89% of their active writing. This number is going to

[4] https://wordcounter.io/blog/how-many-words-does-the-average-person-know/

increase if you also include phrases, but in any case, the table is going to be of a non-trivial size.

The second problem is how to share the code book between the sender and the recipient. You'll see that this is a super common problem in cryptography. Any time you come up with a secret system that requires the sender and the receiver to share information to read or write the messages, you encounter the problem of sharing the information. In this case, the sender and receiver are going to have to share the code book. If the same code is always used then it might not be a problem. But typically, codes are changed frequently just in case an attacker obtains a copy of the codebook. Each time a new codebook is developed it must be securely delivered again.

Changing Languages

The second method for using codes is to translate the message to a different language but retain the same words. For example, changing a message from English to French. Using the previous code example, the phrase **the king is in the castle** becomes **le roi est dans le chateau**. The beauty of this system is that it doesn't require a translation table. That is, it doesn't require a table if both the sender and the receiver are able to understand both languages. The problem with using this type of code is that it's only secure if an attacker doesn't recognize the second language.

And what's the chance that an attacker will recognize your code language? In most cases I'd say it's pretty high, especially in modern times. But, one of the great stories of World War II involved some heroic members of the Navajo tribe who volunteered to be do this type of language translation. Their story is chronicled in the book **Code Talker: A Novel About the Navajo Marines of World War Two** by Joseph Bruchac, and dramatized in the movie Wind Talkers. You can also watch the documentary:

If you read the books or watch the movies you'll see that the code they used actually combined changing words and phrases with translating between Navajo and English, and it was very effective in preventing the Japanese from translating the messages.

Ciphers

Ciphers are the second main branch of cryptography, and they make up the bulk of the field. That is modern cryptography deals primarily with different ciphers. For this reason, we're going to go into quite a bit more depth on ciphers. Ciphers are different than codes as they replace single characters with different characters, as opposed to codes which replace entire words or phrases[5]. This is a key point to remember, as sometimes the distinction between ciphers and codes can seem a little fuzzy. Just remember, ciphers change characters while codes change

[5] https://www.khanacademy.org/computing/computer-science/cryptography/ciphers/a/ciphers-vs-codes

words and phrases. And although it's might seem like it's just a matter of semantics, it's important that you understand the distinction.

You'll start by learning some terms and then look at the two general ways that text ciphers work. After that, you'll learn the details of several historical ciphers. These text ciphers are the first step in the evolution that has led to modern encryption, which means it's important to understand how they work as this foundational knowledge foundation for understanding the algorithms and methods used in modern encryption.

And here are two last things to note before we jump in and start looking at ciphers. The first is that the historical ciphers are all simple enough that you can use them to encrypt text by hand, but as you'll see they were and still are fairly ingenious and can be complex enough to protect messages from any attacker who doesn't have a computer and has to decipher the message manually. The second key point to note is that these manual ciphers work on text characters as opposed to bit-wise data. That is, you can use them to protect text messages, but you can't use them to protect image data files or video files, etc.

Terminology

Here are some terms you will need to know as we start discussing the various methods for encrypting text using ciphers.

Plaintext – This is an unaltered message before it's been encrypted. It's readable by anyone.

Ciphertext – This is a message that has been altered by an encryption scheme to be unreadable by anyone except the intended recipients

Key – This is a special piece of data used to encrypt and decrypt messages. This is important because there really aren't that many cryptographic systems in use, either historic or modern systems. And the details of how all the encryption algorithms work is common knowledge. However, most cryptographic systems will use some special data, called a key, to vary the encryption process. If data has been encrypted with a key, then the same key must be used to decrypt or read the data.

Keyspace – The total number of possible values of keys for a cryptographic algorithm. The size of the keyspace provides an indication of how hard it would be to crack a message by trying to decrypt it using every possible key.

Kerckhoffs's Principle[6] - Auguste Kerckhoffs was a cryptographer who summed up the importance of keys and keyspace when the cryptographic algorithms are well known. His principle states "A cryptographic system should be secure even if everything about the system, except the key, is public knowledge." While the principle may seem obvious today, it's been important enough that it's still pointed out as an important concept, and it's often the subject of questions on certification exams.

[6] http://www.crypto-it.net/eng/theory/kerckhoffs.html

Does Using the Correct Terminology Matter ?

As you're starting to learn, cryptology, like any subject, has its own specific terms like cryptanalysis, steganography, encryption, and hashing, codes and ciphers, etc. Before you learn about these terms and the cryptologic algorithms it can be easy to get confused. In fact, many people often confuse terms like cryptology and cryptography, or code and cipher; or think that passwords are encrypted.

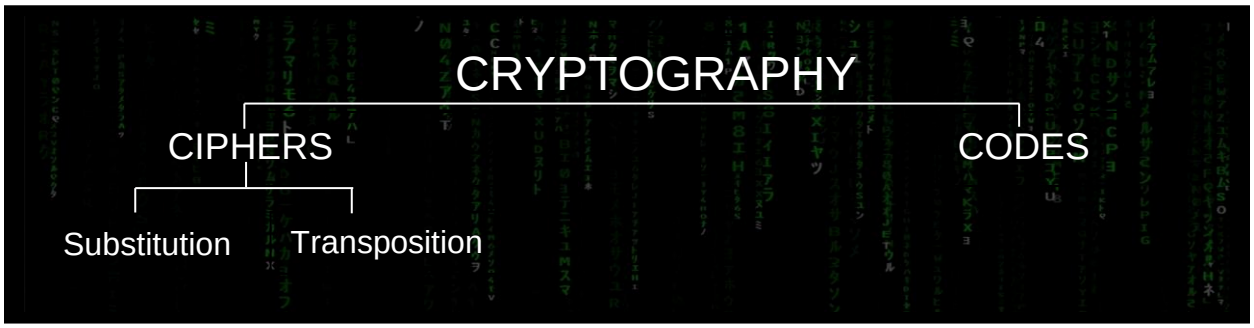
Mixing up terms and possibly using them incorrectly might be ok if you're not working as a professional. It would be like confusing butter and margarine, which isn't a huge problem when you're speaking with most people. But it would be a big mistake if you're talking to a professional chef. Or you might know that the tibia and fibula are bones in your lower leg, but not really know which is which. Most people won't judge you if you don't know the difference. But I know I'd be worried if I needed surgery for my leg and the surgeon kept confusing the two bones.

The Two General Methods used in Text Ciphers

When you first encounter the different text ciphers it can be easy to get lost in all of the details, but what I'd really like you to take away from this section is the fact that all of the text ciphers really only use two general methods for changing a text and making hard to read. That is, while it might seem like there are dozens of methods for enciphering text, there are really only two main methods.

1. **Substitution** – This is where each character in the plain text is swapped for a cipher text character. The swapping is based on some plan or conversion table. For example, every plain text **A** character may be swapped for the cipher text **G** character. Decrypting the message is a matter of reversing the substitutions.

2. **Transposition** – This is where plain text characters remain unchanged during the conversion to cipher text. What does change is the position of the plain text characters. For example, the first character from the plain text message may be moved to the 8th position in the cipher text. Decrypting the message is a matter of moving the characters back to their original positions.



Substitution Ciphers

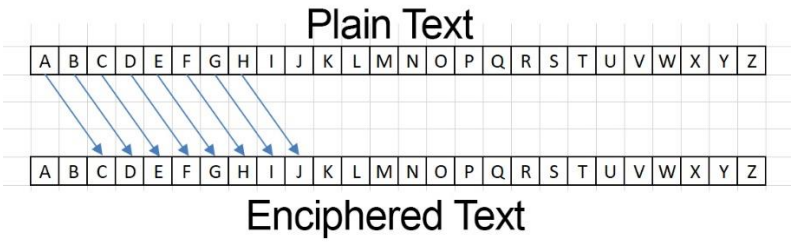
The first type of ciphers we'll look at are the substitution ciphers. You'll start by learning about a special group of substitution ciphers called rotation ciphers, then move on to general substitution ciphers, and finish by learning about translating characters.

Rotation Ciphers

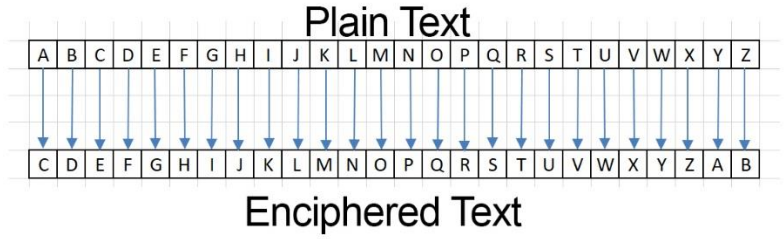
The first set of substitution ciphers you will learn about are **rotation** ciphers. They get their name because they encrypt each plain text character by rotating or shifting it through the alphabet by a specified number of characters. For example, a rotation of three would take each plain text character and shift it by 3 characters to determine the cipher text character. Using a rotation of 3 maps a plain text **A** to a cipher text **D**, and a plain text **B** to a cipher text **E**, etc. Using a rotation of 7 would shift each plain text character by 7 places, and map **A** to a cipher text **H**, and a plain text **B** to a cipher text **I**, etc.

A shorthand way to refer to specific rotation ciphers is to the three characters ROT followed by the rotation number. For example, ROT11 means a rotation cipher that shifts characters 11 places.

One way to envision the rotation is to look at a table that shows the input or plain text on the top row, and the output or cipher text on the bottom row. For example, the following figure shows a shift or rotation of 2 characters. Note that characters at the end of the alphabet will wrap around and start over again at the beginning of the alphabet. In this example **Y** would map to **A** and **Z** would map to **B**.

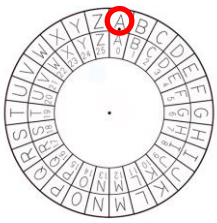


Or here is the same table, but the enciphered text characters, the characters in the bottom row, have been shifted so that they're directly under their corresponding plain text character. This makes it a little easier to see how the characters at the end of the alphabet are mapped.

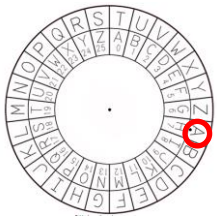


Another way to visualize the rotation ciphers is to look at the famous decoder ring. The decoder ring has a copy of all the characters in our alphabet on an inner ring, and a copy of the same characters on an outer ring. To initialize the process both rings are lined up so that the characters on the inner and outer rings are aligned. That is, **A** on the outer ring is adjacent to **A**

on the inner ring. The next step is to decide how many characters to rotate for the cipher and rotate the outer ring this many characters in the counterclockwise direction. The example in the figure uses a rotation of 7, so that **A** on the outer ring ends up over **H**.

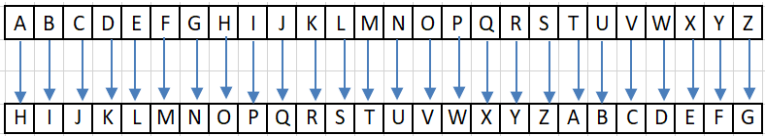


Starting position



Rotate outer ring 7 places

Once the outer ring is rotated the plain text message can be enciphered by finding each plain text character on the outer ring and converting it to the cipher text character on the inner ring. Let's do an example, using a shift of 7 as shown in the figure to encrypt the plain text message **I am the walrus**. While the rings make easy work of setting up the ciphers, I find that using a table makes the actual encryption and decryption a little easier.




```
I - P
A - H
M - T
T - A
H - O
E - L
W - D
A - H
L - S
R - Y
U - B
S - Z
```

Or: P HT AOL DHSYBZ

Messages encrypted with a rotation cipher are decrypted by reversing the process. That is, each cipher text character can be deciphered by finding the character on the inner ring and converting it to the plain text character on the outer ring.

Caesar Cipher – ROT3

There's one famous rotation cipher, ROT3, which is also known as the Caesar cipher. It's not famous because it's any stronger or better than any of the other rotations, it's famous because it

was used by Julius Caesar to keep his salad dressing and pizza recipes a secret. LOL. It actually was used by Julius Caesar, even though I doubt he was the inventor. It's not important that you know this trivia, but it may come in handy if you ever take a certification test. I've seen several questions where they ask about ROT3 or the Caesar cipher and assume you know what they are.

Technical Characteristics - Rotation Cipher

Here are some of the technical characteristics of rotation ciphers. These are things we can use to compare them to other ciphers and codes, and to help you make a little more sense out of the terminology.

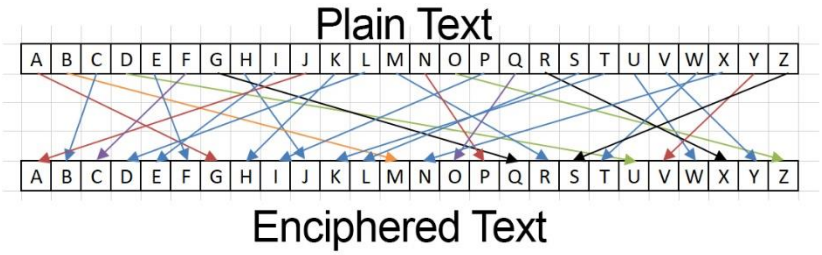
The first characteristic is the key, or what information needs to be exchanged between the sender and the recipient so the message can be decoded. For rotation ciphers the only thing that the recipient needs to know is how many characters to shift or rotate. For rotation ciphers this single number is the "key" or piece of information required to customize the encryption/decryption algorithm.

The keyspace characteristic of rotation ciphers is the number of possible keys or variations. In the case of the English alphabet there are only 25 possible variations. This assumes that we're only using a single case, that is just upper case or just lower case; and that we're never going to use a rotation of 26 where we'd map a character back to itself.

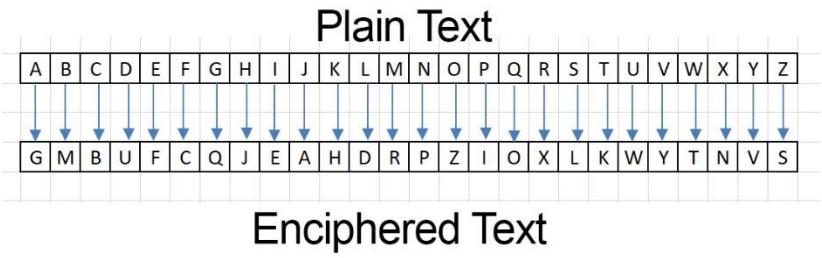
The last important characteristic of rotation ciphers is how to break them. While breaking a message encrypted with a rotation might seem difficult at first, there's an attack called frequency analysis that makes it pretty simple to accomplish. You'll learn the details of frequency analysis later in this chapter, but for now let's just say that there are certain letters and letter combinations that occur more frequently than others that provide strong clues that can be used to crack messages.

General Substitution Ciphers

Now it's time to look at general substitution ciphers. A general substitution cipher functions much like a rotation cipher, the difference is that instead of rotating or shifting the same number of characters to encipher every character, the character mappings can occur in ways that don't meet any specific pattern. Here's an example of a mapping between plain text characters and cipher text characters that makes it easy to visualize what a general substitution cipher may look like.



Here's another view of the same substitutions that's should be easier to read, and much easier to use if you had to encipher or decipher text by hand.



Using the above table, the plain text message **I am the walrus** would be enciphered as:

```
I – E
A – G
M – R
T – K
H – J
E – F
W – T
A – G
L – D
R – X
U – W
S – L
```

Or writing just the enciphered text and adding the spaces back in it would be:

E GR KJF TGDXWL

Technical Characteristics - General Substitution Cipher

Here are some of the technical characteristics of substitution ciphers. The first is defining the key. In this case the key is the table that shows how to map between the plain text characters and the cipher text characters. This is a little more complicated than the rotation cipher key, but it's still not a lot of information to exchange. If we stick with all upper-case or all lower-case letters only then the table would be 2 rows of 26 character. Or you could just exchange one row of 26 characters, if you assume the characters in the missing row are in alphabetical order.

If we add more characters to the table, like both upper and lower-case, or numerals and special character, the table will become larger and more complicated. But even if we include every alphanumeric and special character in the table there will be less than 100 characters.

The second technical characteristic is the number of possible variations, or the "keyspace" for substitution ciphers. In this case, we have to calculate all of the possible ways to do the mappings. For the 26 character English alphabet the short way to do this is to realize that it's equal to 26! which equals 4.032914611266056e+26 or ~ 403000000000000000000000000.

Here's the longer explanation if you don't see the logic behind this, or if you don't remember how factorials work in math. The first step is to determine how many ways there are to map a single character. Let's start with **A** to keep things simple. We can map **A** to any of the 26 other characters. Of course, it wouldn't make a lot of sense to map it to itself, but it's still a possibility. For our example, let's say we map **A** to **K**.

Next let's look at how many ways there are to map the letter **B**. We can map it to any letter except "k", because we're using that for the letter **A**. This means that there are 25 possible mappings. If we now look at the total possible ways to map our first character **A** and **B**, our second character together, it would be 26 possibilities for **A** times 25 possibilities for **B**. That is, we can choose any character mapping we want for **A** but that will reduce the number of possible mappings for **B** by one.

We can continue this for the other characters. For example, the letter **C** can be mapped to any character except the ones we chose for **A** and **B**, so the number of possible mappings would be reduced to 24, and the total number of ways to map three characters would be 26x25x24.

If we continue this all the way through the character **Z** we see that the total possible mappings for all the characters would be:

26x25x24x23x22x21x20x19x18x17x16x15x14x13x12x11x10x9x8x7x6x5x4x3x2x1

This sequence of multiplications happens often enough in math that it has been given its own name. It's called a factorial, and it's denoted by adding an exclamation point, or bang, after the starting number in the sequence. For example, 4x3x2x1 is written as 4**!**

In any case, the number of possible character mappings is pretty huge. Even though 26! might not look like a large number, if you do the calculations you'll find that:

26! = 403,291,461,126,605,635,584,000,000 or 403 septillion

Remember that we call this the keyspace of the total number of possible variations in mappings. Also note that this is the keyspace if we limit the text to 26 characters, and allow for the possibility of mapping a character onto itself. If we disallow mapping characters onto themselves the number of variations drops to 25! or 15 septillion, which is still a pretty large number. If we use more characters, like numerals, or both upper case and lower-case characters the keyspace will be even larger.

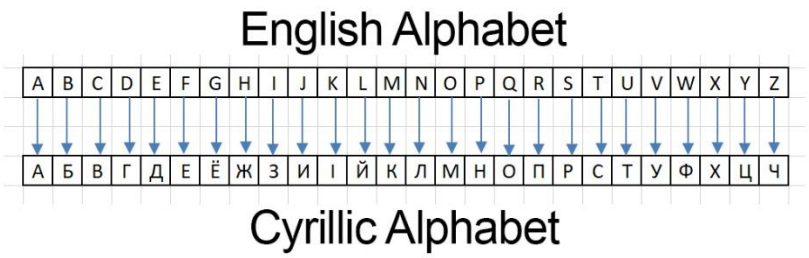
Because the keyspace is so large it might lead you to believe that substitution ciphers would be hard to crack. And they would be hard to break your only approach was to try and brute force a solution by trying every possible mapping. However, as you'll see in a minute, it turns out that substitution ciphers are almost childishly easy to break using frequency analysis. This goes to show that the size of the keyspace isn't the best indicator for an encryption scheme's strength.

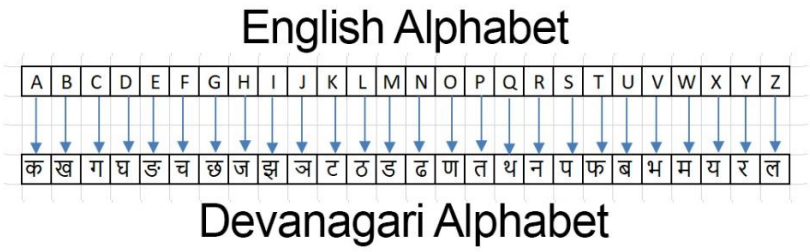
Translation - Changing Alphabets or Character Sets

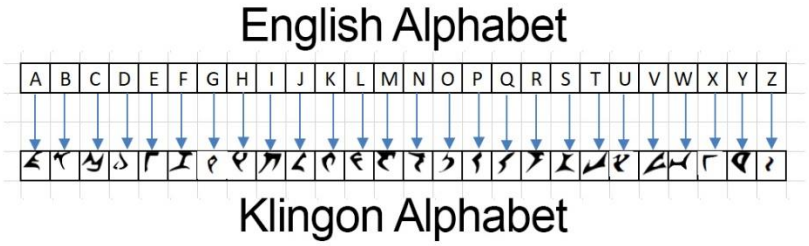
All of the substitution ciphers we've looked at so far map plain text characters from an alphabet to cipher text characters in the *same* alphabet. Translation is another form of substitution, but in this case each plain text character is mapped to a cipher text character in a *different* alphabet or character set

Those of you who have been paying attention may be thinking that translating is a code, not a cipher. It would be a code if we were translating entire words, but in this case, we're just mapping characters from one alphabet to another. The following tables show the character mappings for translating characters from the English alphabet to characters from a few other

alphabets. The first is Cyrillic, which is used in many Eastern European languages such as Russian. The second is Devanagari, which is used in India, and the third is Klingon for my nerdy Star Trek friends.







(For those of you detailed minded people here are a few notes on the substitution tables:

1. I only listed the first 26 characters of each alphabet even though Cyrillic and Devanagari have more than 26. This is because this isn't a language class, the purpose of the table is to provide an example of encoding characters from one alphabet to another.

2. Those of you who know Cyrillic may disagree with some of the character positions. For example the Cyrillic **Д** should be mapped to the English character **D**, and the Cyrillic characters **P**, **C** and **T** should align with **R**, **S** and **T**. But as you can see, I simply added the Cyrillic characters in alphabetic order, so many of the characters that are phonetically equal aren't aligned in the table.

Once again, this isn't a language class, so try not to worry about this too much.)

To demonstrate this, here's the characters from **I am the walrus** enciphered using the different alphabets:

Cyrillic – З АК СЖД ФАЙПТР

Devanagari - झ क ड प ज ङ म क ठ न ब प

Klingon - 

To demonstrate the difference between *encoding* the message, where we encode plain text words into cipher text words and *enciphering* where we swap individual characters, let's look at the phrase **I am the walrus** again.

If we use the English – Cyrillic substitution table to encipher each character we get the following

I – З
A - А
M - К
T - С
H - Ж
E - Д
W - Ф
A - А
L - Й
R - П
U - Т
S - Р

Or writing just the enciphered text and adding the spaces back in it would be:

З АК СЖД ФАЙПТР

If you showed this encrypted phrase to a Russian speaker it would look like nonsense. But if we encoded **I am the walrus** by swapping the English words for the correct Russian words and grammar the encoded phrase would be:

я морж

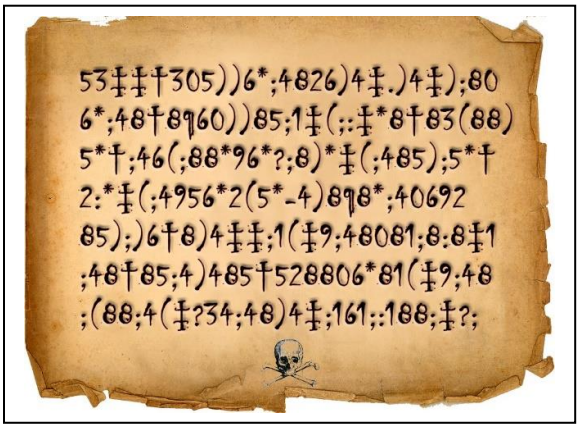
As you can see the encoded text and the enciphered text are much different.

Obviously changing languages has some limitations. The first is it requires the use of two different character sets. This would pose a problem for two languages that share character sets, like English and most of the western European languages. For example, translating English characters to Spanish or Italian characters wouldn't be very helpful because all these languages mainly use the Roman/Latin character sets. In other words, the plain text and the encrypted/translated text would be identical or nearly identical.

The second limitation is that decrypting messages can be done by anyone who knows both languages. For most human languages makes this method very insecure since there are typically many people that can read and write any two languages you choose.

The Gold Bug Cipher

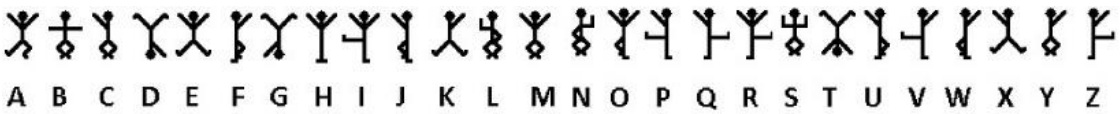
This type of cipher might be a little more secure if you just make up a character set, like Klingon[7]. This was actually used in the plot for several famous cases or stories in the 1800's. For example, Edgar Allan Poe wrote a short story called "The Gold Bug" where the clues to a buried treasure were encrypted using a made-up alphabet that's now known as the Gold Bug Cipher. Most of the cipher text characters Poe chose were "normal" English numbers and punctuation characters, but he threw in a couple like ‡ and ¶ to make it look a little more exotic. The following table shows the substitutions Poe used in the Gold Bug cipher.



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
5	2	-	†	8	1	3	4	6	,	7	0	9	*	‡	.	$	(	)	;	?	¶	]	¢	:	[

The Dancing Men

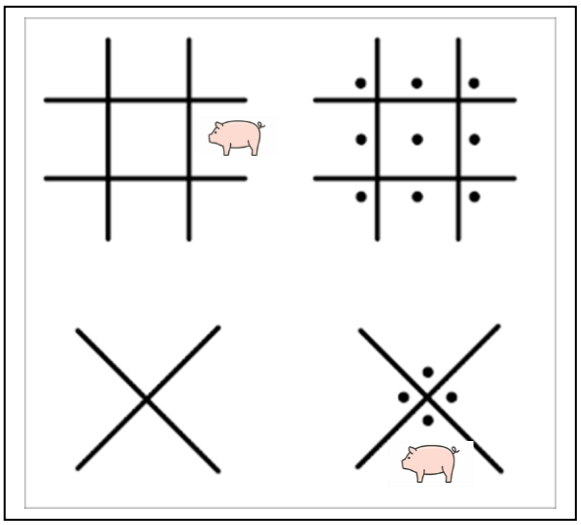
Sir Arthur Conan Doyle used a cipher called the Dancing Men in one of his own favorite Sherlock Holmes stories. This alphabet used characters that looked like, yes you guessed it, dancing men. Text written in this alphabet looks pretty cool and pretty crazy, but I imagine it would be difficult to write a message of any length using these characters. Speaking for myself, I'd have a hard enough time remembering the characters, much less drawing the intricate shapes for messages of any length.



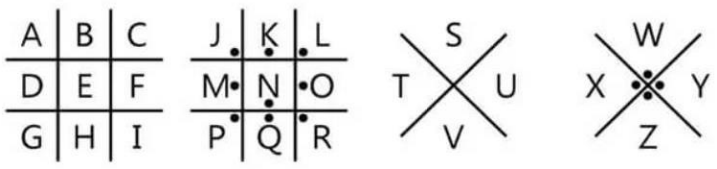
[7] I'd guess I'd better apologize and correct myself before I get any hate mail from you Trekkies. I realize Klingon was a fictional language when Star Trek first came out. But there are now people who use it as a real language.

The Pigpen Cipher

The Pigpen cipher is another ingenious variation of using a custom character set. It was used by the Freemasons, who are/were the evil arch villains in many conspiracy theories but heroes in many others. Go figure. In any case the cipher “alphabet” is built by first drawing the shapes, or pigpens using a set of straight lines and dots as shown in the figure. Think of each section or space between lines as a pen which will hold a single pig.



The pigs are then put into the pens, one pig per each pen, although the pigs are actually letters. This results in the following.



Enciphering a plain text character is done by drawing the shape, the lines, and dots, of the portion of the pigpen holding the plain text character but leaving the actual text character out.

Using the Pigpen cipher the plain text phrase **I am the walrus** would be enciphered as:

The beauty of the Pigpen cipher is in its simplicity. All you need to remember to rebuild the translation table is how to draw the pig pens. And they’re just two tic-tac-toe grids followed by two big X shapes, with dots added to the second tic-tac-toe grid and the second X.

Compare this to building the translation tables used by the Gold Bug and Dancing Men ciphers. Building the Gold Bug and Dancing Men tables requires memorizing the characters and getting them in the correct order. I don’t know about you, but it’d take me quite a bit of time and work to be able to rebuild either of these tables, especially the Dancing Men since the characters are all different than anything I know.

Technical Characteristics - Changing Alphabets or Character Sets

It's not critical that you're able to rebuild and of these ciphers from memory. But what you do need to take away is that of encrypting text by translating it from one character set to another is really just a variation of a substitution cipher. The key, or information that the sender and recipient need to exchange, is the translation table. The keyspace may be the same magnitude as for regular substitution, or it might be argued that it's much smaller. That is, for ciphers like the Dancing Men or the Gold Bug the character mappings never change so the keyspace may be of size 1 rather than 26!

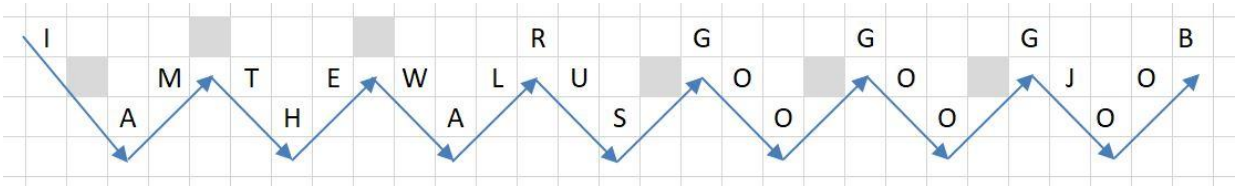
Breaking these types of ciphers is surprisingly simple. The encrypted messages may look exotic and intimidating at first, but like all substitution ciphers they're actually very simple to break using frequency analysis. It doesn't matter which alphabet is being used for the cipher text, the frequency of the characters in the plain text will remain the same. So, as you'll soon learn it's simple to determine the mapping between cipher text and plain text characters.

Transposition Ciphers

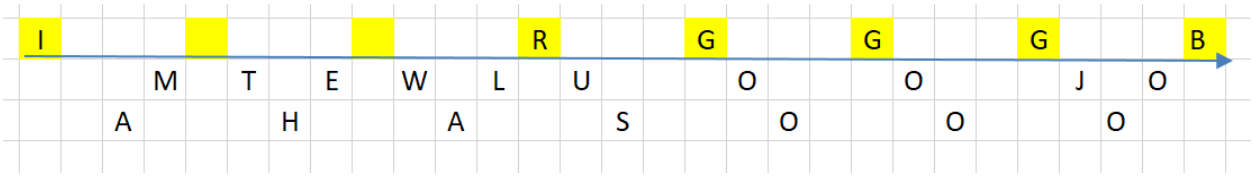
The next big class of ciphers are transposition ciphers. The identifying characteristic of transposition ciphers is that they don't replace any of the plain text characters, they just reorder the characters, so they end up scrambled in the cipher text. There are several different variations of transposition ciphers, such as the rail fence, route ciphers, columnar transposition, and even the Scytale that you learned about previously can be recreated with a transposition cipher.

Rail Fence

To help visualize what transposition is, let's look at a famous transposition cipher called the Rail Fence cipher. The first step in the Rail Fence cipher is to pick a number that will determine how many lines or rails the fence will have. This number must be larger than 1, but not too large unless you have a lot of text and a big sheet of paper to work with. You'll see why you don't too large a number in a few seconds. For this example, I'm going to use 3 lines. Next, write the plain text in a zig zag pattern across different lines. The following figure shows how the lyrics from "I Am the Walrus" would be written. Note that in the figure the gray boxes are used to indicate a space character.



The final step is to write the cipher text by writing all the characters in the first row, followed by all the characters in the second row, and finally all of the characters in the third row. The figure shows that the first row would be **I RGGGB**.



Writing the text from the first row, followed by the text from the second and third rows would change the entire lyric:

> I am the walrus
> Goo goo gjoob

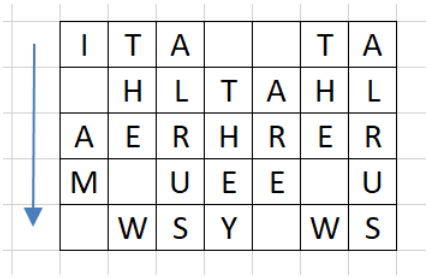
To:

> I RGGGB MTEWLU O O JO AHASOOO

The main concept to notice with the rail fence cipher, and any transposition cipher, is that the plain text characters aren't changed, they're just scrambled and written in a different order.

Route Ciphers

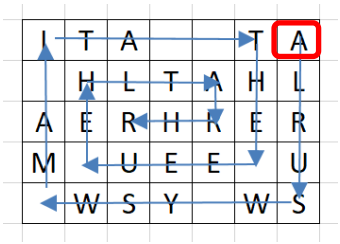
The route cipher works is another transposition cipher. The first step in encrypting with a route cipher is to decide the number of rows to use in a grid which will be used to enter the plain text. After drawing the grid, the plain text is added by either writing it across the rows or down the columns. In the example in the figure, the text is added writing the first character in the upper left corner, then writing subsequent characters down the first column. When the first column is filled continue writing the plaintext at the top of the next column. The following example shows writing **I am the walrus they are the walrus** in a grid with 5 rows.



I	T	A			T	A
	H	L	T	A	H	L
A	E	R	H	R	E	R
M		U	E	E		U
	W	S	Y		W	S

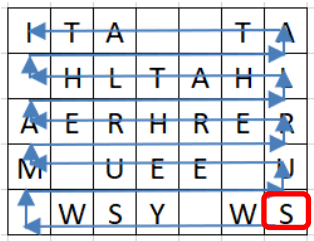
Once all the plain text has been written the next step is to decide a route for writing the cipher text. Here are some example routes:

1. **Inner Spiral** - Starting at the upper right corner and spiraling inward in a counterclockwise position.



Following this route creates the following cipher text: **ALRUSW YSW MA I TA THE EEU EHLTARHR**

2. **Upward Zig Zag** - Starting at the lower right corner and writing all the characters in the last row as you move to the left corner. After you write all the characters in a row, continue by moving up to the right most character in the row above.



Following this route creates the following cipher text: **SW YSW U EEU M RERHREAIHATLH AT ATI**

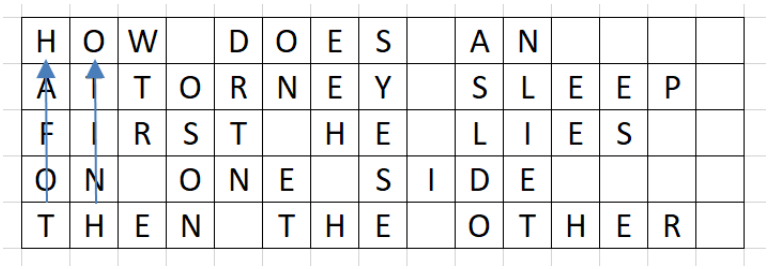
As you can see there are quite a few different routes that could be used. If someone wants to build the decryption system they'll need to know the grid size, whether the text was written across the rows or down the columns, the route and the starting point.

Scytale

If we go back and look at the Scytale you'll see that it's really a route cipher and you don't need a baton or spear to encrypt your message. As an example, I've built a 5 row grid and added the same plain text that I used on the scytale. The plain text is written in normal left to right, top to bottom order as shown in the following figure.

H	O	W		D	O	E	S		A	N				
A	T	T	O	R	N	E	Y		S	L	E	E	P	
F	I	R	S	T		H	E		L	I	E	S		
O	N		O	N	E		S	I	D	E				
T	H	E	N		T	H	E		O	T	H	E	R	

The route for the cipher starts at the bottom left corner and moves up the column. When the last letter in the column is reached, the route continues with the bottom character in the next row to the right.



This results in the following cipher text: **TOFAHHNITOE RTWNOSO NTRDTE NOH HEEESESY I ODLSATEILNH EE E SE R P**

Technical Characteristics – Transposition Ciphers

There are dozens of variations of transpositions ciphers, but they generally share the same technical characteristics. The key for a transposition cipher will have at least two parts, the grid size and the route. (One of the variations that we didn't look at called columnar transposition has a keyword that also must be exchanged.)

There's not really a good way to assign a number to the keyspace with route ciphers, at least not as cleanly as the keyspace definition for substitution ciphers. I'm sure some smart mathematician could come up with a formula. This is, given a certain number of plain text characters there would be only so many ways to divide them into rows and columns, and then with this grid there would be a way to calculate the possible number of routes.

But in any case, the size of the keyspace doesn't really matter because route ciphers have a huge weakness that make them easy to break. Since the plain text characters are not changed as they're moved to the cipher text, performing a frequency analysis will make it obvious that the plain text characters have simply been rearranged. This is a huge tip off to the cryptanalyst that they're probably dealing with a transposition cipher. At this point they can attack the cipher text by doing something called anagramming, which is sliding chunks of text around until words start to appear.

Cryptanalysis

The next thing you'll learn about is cryptanalysis, which is the science of breaking encryption schemes. As you probably guessed, this field developed about as people intercepted encrypted messages and wanted to know what they contained. Imagine that your country is being invaded and you intercept the following message:

> O YPPS PFPDBL CPGQPLM RPYPFLP OL MIP CPLM CGSS. O COSS YGQP GF GMMGEQ MA MIP PGLM GM MCA GD, CGOM LPXPF DOFWMPL MIPF DGQP GF GSS AWM GMMPDHM AF MIP CPLM.

Can you look at this and determine what type of cipher was used, and more importantly what the decrypted message says?

If you were a cryptographer and created your own substitution ciphers you might think to try and brute force an answer by trying every possible character mapping. But if you remember there are 25! or 26! possible mappings. The brute force approach would work eventually. But if you were trying each solution by hand there's no way you could try all the possible variations in a lifetime, much less in time to use the information.

Reading encrypted messages was a real problem, and as the saying goes "necessity is the mother of invention". Very intelligent people began working on a process for decrypting messages and in the 9th century an Arabian scholar named Al-Kindi provided the first known recorded explanation of a method for cracking substitution ciphers. Al-Kindi didn't give method a name, but it has since become known as frequency analysis. The method has this name because it uses statistics regarding the number of times, or frequency, that individual characters, words, and certain letter patterns appear in the cipher text of a message.

Here are main facts and observations that make frequency analysis possible, as well as the general steps in using frequency analysis to crack a message encrypted with a substitution cipher:

1. In any language, English included, certain letters appear more frequently than others. And in fact, the number of times each letter appears has been measured and analyzed. These studies show that the individual letters occur at very consistent frequencies.

   The chart to the right shows the frequencies for each letter in English writing. The chart was developed using data from Google Books Ngrams[8] (Source 1), which counted ~3 trillion letters, Leipzig University[9] (Source 2) which counted letters in 4.5 billion characters of English text, and Cornell University[10] (Source 3) which counted the number of times each letter showed up in 40,000 different words. The total number of times each letter appeared was then divided by the total number of all the letters combined. This percentage shows how frequently the letter appears in relationship to the other 25 letters. (You may recognize Leipzig as a city in Germany and be worried that their data shows letter counts from German words. Luckily, they worked on a corpora or body of text, that was written in English.)

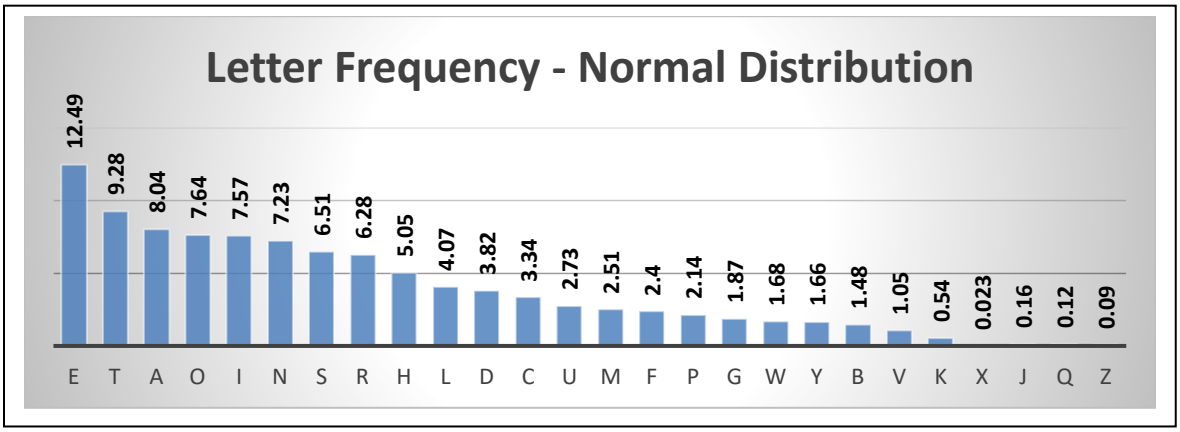
Letter	Source 1 Frequency	Source 2 Frequency	Source 3 Frequency
E	12.49	12.1	12.02
T	9.28	8.94	9.1
A	8.04	8.55	8.12
O	7.64	7.47	7.68
I	7.57	7.33	7.31
N	7.23	7.17	6.95
S	6.51	6.73	6.28
R	6.28	6.33	6.02
H	5.05	4.96	5.92
L	4.07	4.21	3.98
D	3.82	3.87	4.32
C	3.34	3.16	2.71
U	2.73	2.68	2.88
M	2.51	2.53	2.61
F	2.4	2.18	2.3
P	2.14	2.07	1.82
G	1.87	2.09	2.03
W	1.68	1.83	2.09
Y	1.66	1.72	2.11
B	1.48	1.6	1.49
V	1.05	1.06	1.11
K	0.54	0.81	0.69
X	0.023	0.19	0.17
J	0.16	0.22	0.1
Q	0.12	0.1	0.11
Z	0.09	0.11	0.07

   The data is sorted so the letters that appear more frequently are at the top of the table, while those that appear less frequently are at the bottom. The data shows that letters like **E** and **T** are much more likely to occur than other letters such as **Z** and **Q**. As you can see the exact frequencies may vary slightly, especially for the letters that don't appear often. But in general, the outcomes were very close. And we don't need exact numbers for our purposes, so knowing the exact percentage doesn't really matter. What matters is that letters like **ETAOIN** are much more likely to appear than others, with **E** the most likely.

   Looking at an individual letter, the data shows that the letter that occurs most frequently is **E**, and it's over 12 times as likely to appear as other letters, while the letter **T** is around 9 times more likely to appear than other letters.

[8] https://norvig.com/mayzner.html
[9] D. Goldhahn, T. Eckart & U. Quasthoff: Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages.
In: *Proceedings of the 8th International Language Resources and Evaluation (LREC'12), 2012*
[10] http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

Here's another view of the data that makes it easier to visualize the letter frequencies.

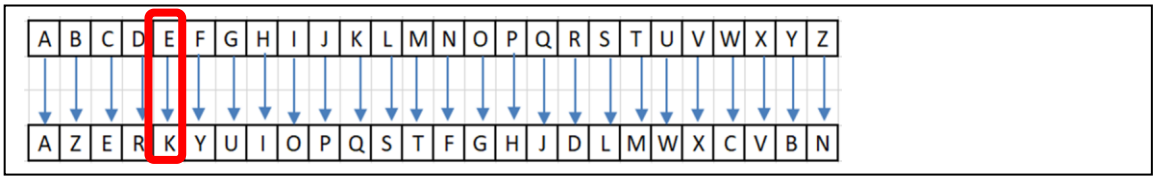


A crucial observation to take away from this data, is that in any plain text document, the individual letters will almost always appear in frequencies close to those shown in the table. In longer documents with more text the frequencies should be very close, but in short documents the frequencies may vary. That is, if you check a short message with only 10 characters, the letter frequencies will be way off from the normal distribution, is you check a message with 1000 characters the frequencies should be closer to the normal distribution, and if you check a message with 10,000 characters the letter distribution should be very close to the normal distribution.

The next thing to do is make a connection between the letter frequencies and cipher text created by a substitution cipher. Can you see how the letter frequencies can be used to help decrypt the message, even if the plain text letters have been changed to cipher text letters?

Remember that with a substitution cipher there's going to be a one-to-one mapping between plain text letters and cipher text letters. This means that all the plain text **E** characters are going to be mapped to a single cipher text character. And that means whatever character that is, should appear ~12 time more frequently than other characters in the cipher text. This also means that there's an excellent chance the letter that appears most frequently in an encrypted message can be decrypted back to the letter **E**.

For example, if we use a substitution cipher that uses the following map, the letter **E** is mapped to the letter **K**. So, if we count the number of times each letter occurs in a piece of cipher text created with this map, the letter **K** would most likely occur more than any other letter in the cipher text.

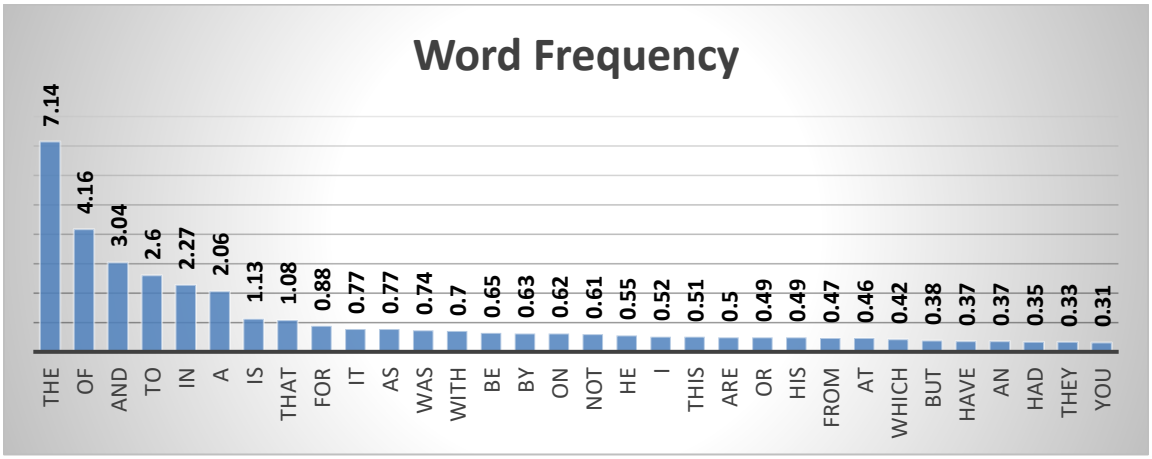


The first step in performing a frequency analysis is just counting the number of times each letter occurs in the cipher text. After that, we try to replace the letters that occur most frequently in the cipher text with **E**, **T**, **A** etc.

If you're good at the game Wheel of Fortune you might be able to completely decrypt some of the words after making replacing just these three characters. But if not, don't worry as frequency analysis can still provide additional clues.

2. The next step in frequency analysis looks for clues from individual words instead of looking at the individual characters. Not surprisingly, just as some letters appear more than others, some words appear much more often than others. Peter Norvig, while he was a director at Google performed an analysis[8] on how often individual words appear in all the books digitized by Google. The analysis counted ~750 billion words in total and found ~100,000 unique words. It also showed that some words were used much more frequently than others.

   The following chart shows the most frequently used words.



   Applying the word frequency to the short words in an encrypted message can reveal a lot of clues regarding the encryption scheme. Here are some tips for using this information to decrypt words with 1, 2 or 3 characters.

   a. **1 character words**. How many words can you think of that only contain a single letter? The words **I** and **A** are pretty much it. Of course, there could be some single letter slang words like **K**, but the only "officially" recognized words are **I** and **A**.

      This means that any single letter words in a cipher text message must decipher to either **I** or **A**. You could assume that single character words are more likely to be an **A** based on frequency analysis. But it's possible that **A** has already been decrypted since it's one of the most frequently used letters. If you've already decrypted **A** then the single character words must decrypt to **I**.

b. **3-character words**. The next easiest words to guess are 3 character words. It might seem a little strange that 3 character words are easier than 2-character words, but there are two words, **the** and **and,** that occur so frequently they're typically easy to spot in the cipher text. Here are a few clues that give them away:

   - In most cases the letters **T**, **E**, and **A** have already been deciphered. This means that if you see 3-letter cipher words with the pattern **T_E** the middle letter almost certainly deciphers to **H**.

   - And likewise, if you see 3-letter cipher words with the pattern **A**_ the last two letters are probably **N** and **D**. This becomes increasingly likely if you see several instances of the pattern.

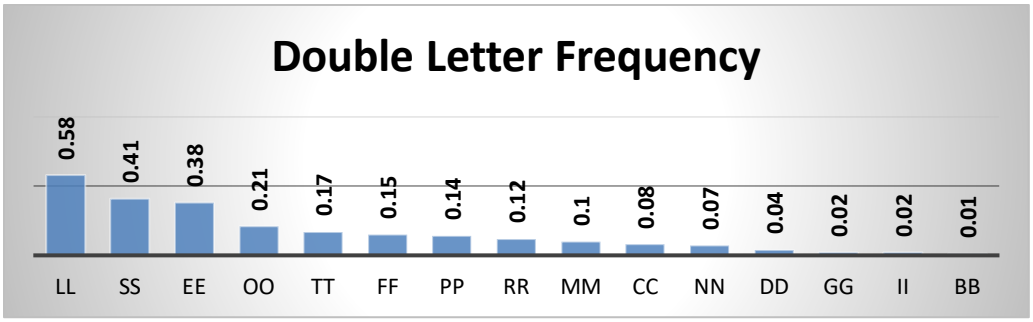
c. **2-character words**. The next clues come from cipher text words with two letters. There are several 2-character plain text words such as **it**, **in**, **if**, **is**, **at**, **an**, **on**, **he**, etc. so finding the plain text isn't quite as easy as it is for 1 or 3 letter words. But if you look for clues from the plain text letters you've already deciphered you can usually make some progress.

3. The last step is to check for patterns of letters within words. The easiest ones to spot are:

   a. **Twins**. These are patterns where the same letter appears twice in a row. As shown in the chart these will most likely decipher to[11] **ll, ss, ee, oo, tt, ff**, etc.

      If the doubled letters appear in the middle of a 4 character word this provides a clue that the letters most likely decipher to **ee** or **oo**. There is a slight chance you might be looking at a word like **anna** or **otto**, which you would see if you've already deciphered the **t** and the **a**. And since you've probably already deciphered **e**, it's a good clue that any unsolved doubled letters decipher to **oo**.

      If the doubled letters aren't in the middle of a 4 letter word then the odds are pretty high that they are **ll** or **ss**, since you've already deciphered **ee**, **oo**, and **tt**; and the frequency of the other doubles are much lower. But you should use the context of the other deciphered letters to guide your decision.



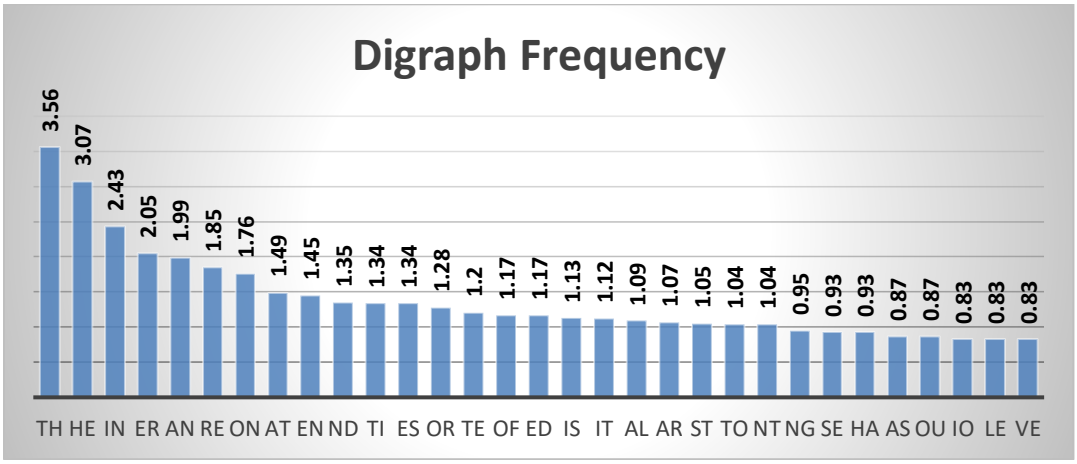
[11] https://blogs.sas.com/content/iml/2014/10/03/double-letter-bigrams.html

b. **Word Endings**. You should also check for the same 2 or 3 character patterns that appear at the ends of several words. If the same pattern appears multiple times, then you can look to the frequency of word endings for clues. The most common word endings are **ed**, **ing**, **ion**, **ist**, **ous**, and **ent**.

   - If you see the same 2 letter ending on several words, and if that pattern starts with an **e**, which should already be deciphered, the other character is most likely a **d** as in **ed**, followed in likelihood by an **r** as in **er**.

   - If you see the same 3 character pattern at the ending of several words and you've already solved any of the other letters such as **e**, **s** or **t** it narrows down your choices. If you haven't solved any of the other letters in the three letter ending you can try **ing**, **ion**, **ist**, **ous**, and **ent**, and see how the plain text letters fit in other word solutions.

c. Usually by this point, solutions for the remaining text characters will start to take shape. If not, you can use the following data from Peter Norvig[8] which shows the most common digraphs or two letter combinations to try and find additional solutions.



Frequency Analysis Demonstration

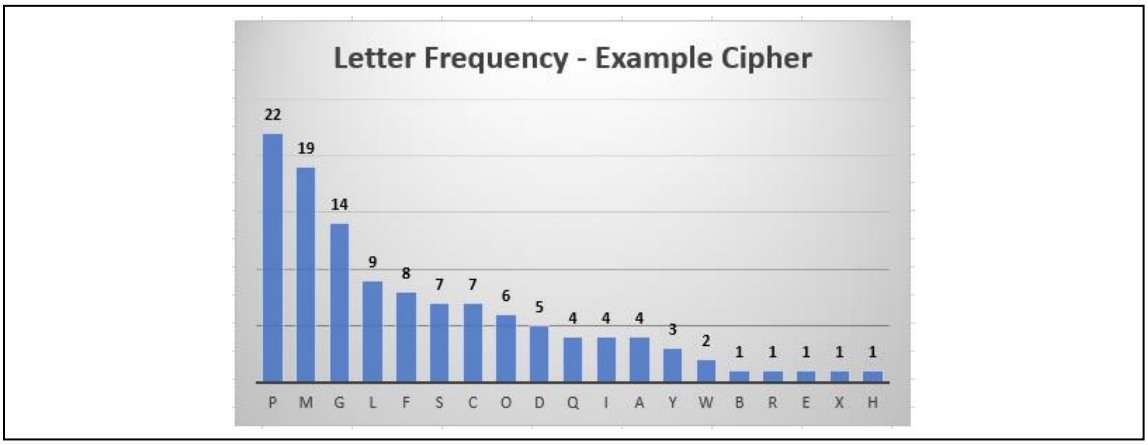
In this section we'll walk through the frequency analysis process and use it to crack this message:

> O YPPS PFPDBL CPGQPLM RPYPFLP OL MIP CPLM CGSS. O COSS YGQP GF GMMGEQ MA MIP PGLM GM MCA GD, CGOM LPXPF DOFWMPL MIPF DGQP GF GSS AWM GMMPDHM AF MIP CPLM.

(I'm going to be honest and warn you that this is one of those things that can be a little hard to follow in written dialog. I think you might have a better learning experience if you just jump right in and try to solve some encrypted messages yourself. There are several online sites you can use that will generate messages with a substitution cipher and provide a nice interface for

helping you solve the cipher. To find these sites do an Internet search for **cryptoquote** or **cryptogram**.)

1. **Run a frequency analysis**. You can count each character by hand, or you can use one of the many online web sites to get a count of each character. Running a frequency analysis on the cipher text produces the following table:



   Note that in the cipher text the letter **P** occurs most frequently, followed by **M** and **G**. This is a strong clue that the we can replace any **P** characters in the cipher text with an **E**. It's also likely that cipher text **M** and **G** are plain text **T** and **A**, although these might be reversed.

   Plugging these plain text characters into the cipher text results in the following. Note that the lower-case letters are the deciphered characters changed back to plain text letters, while the upper-case letters are the cipher text characters.

   O Y**ee**S **e**F**e**DBL C**ea**Q**e**L**t** R**e**Y**e**FL**e** OL **t**l**e** C**e**L**t** C**a**SS. O COSS Y**a**Q**e** **a**F **atta**EQ **t**A **t**l**e** **ea**L**t** **at** **t**CA **a**D, C**a**O**t** L**e**X**e**F DOFW**te**L **t**l**e**F D**a**Q**e** **a**F **a**SS AW**t** **atte**DH**t** AF **t**l**e** C**e**L**t**.

   This might be easier to see if the unsolved characters are replaced with blank spaces.

   _ _ee_ e_e_ _ _ _ea_e_t _e_e__e _ _ t_e _e_t _a_ _. _ _ _ _ _ _a_e a_ atta_ _ t_ t_e ea_t at t_ _ a_, _a_t _e_e_ _ _ _ _te_ tle_ _a_e a_ a_ _ _ _t atte_ _t _ _ t_e _e_t.

2. **One character words**. There are only two occurrences of 1 letter words in the cipher text, both the letter **O**. One letter words must be either **A** or **I**, but since **A** has already been deciphered **O** has to decipher back to **I**. The nice thing about deciphering **O** for **I** in the 1 letter words is that this solves **O** for **I** in several other cipher text words. Putting this in the message results in:

   i YeeS eFeDBL CeaQeLt ReYeFLe **i**L tIe CeLt CaSS. **i** C**i**SS YaQe aF attaEQ tA tIe eaLt at tCA aD, Ca**i**t LeXeF D**i**FWteL tIeF DaQe aF aSS AWt atteDHt AF tIe CeLt.

Or

i _ee_ e_e_ _ _ _ea_e_t _e_e__e i _ t_e _e_t _a_ _. i _ i _ _ _a_e a_ atta_ _ t_ t_e ea_t at t_ _ a_, _ait _e_e_ _ i _ _te_ tle_ _a_e a_ a_ _ _ _t atte_ _t _ _ t_e _e_t.

3. **Three character words**. The three letter words in the cipher text are **tle**, **aSS, AWt** and **tCA**.

    - Hopefully it's obvious that **tle** must be **the**, which means that cipher text **l** must decipher to **h**.

    - The cipher text **aSS** has a set of twins, the doubled characters **SS**. Looking at the frequency chart for double letters indicates that **S** should decipher as either **L** or **S** which would give us either **all** or **ass**. It's possible, but unlikely that **S** was encrypted to itself, and much more likely that **S** decrypts to **L**.

    - **AWt** is a little trickier as there are a lot of three letter words that end in **t** such as bat, not, lot, out, bit, etc. We can throw out a lot of the possibilities since we've already deciphered **a**, **e**, **i**, **t**, **h** and **l**. But it still leaves too many possibilities to make a good guess; so I'm going to save it for now, and return to it after we've deciphered a few more letters.

    - **tCA** could also be deciphered in several possible ways, so I'm also going to save it until we have a few more letters.

    Adding **h** and **l** to the message results in:

    i Yee**l** eFeDBL CeaQeLt ReYeFLe iL t**h**e CeLt Ca**ll**. i Ci**ll** YaQe aF attaEQ tA t**h**e eaLt at tCA aD, Cait LeXeF DiFWteL theF DaQe aF a**ll** AWt atteDHt AF t**h**e CeLt.

    Or

    i _eel e_e_ _ _ _ea_e_t _e_e__e i _ the _e_t _all. i _ill _a_e a_ atta_ _ t_ the ea_t at t_ _ a_, _ait _e_e_ _ i _ _te_ the_ _a_e a_ all _ _t atte_ _t _ _ the _e_t.

4. **Two character words**. There are quite a few two letter words, which have been highlighted in the message.

    i Yeel eFeDBL CeaQeLt ReYeFLe iL the CeLt Call. i Cill YaQe aF attaEQ tA the eaLt at tCA aD, Cait LeXeF DiFWteL theF DaQe aF all AWt atteDHt AF the CeLt.

    - **iL** – There's a good chance this could be **if**, **in**, **it**, **is**. And there's a very slight chance it could be **id**. Since we've already deciphered **t** it narrows the choice down to **if**, **in** or **is**. The letter **n** has the highest frequency so let's try that first. We can check by replacing **L** with **n** in other words such as **eaLt**. This would decipher to **eant**, which isn't a word, so we know **L** is not **n**. Next, let's try replacing **L** with **s**, which changes **eaLt** to **east**. Since east is a word, this looks like a match.

- **tA** – The only letter **A** could decipher to is **o**. That is, **A** must be a vowel, and **o** is the only vowel that makes a word. Ok, **ti** is a word used by musicians, and **ty** might be an abbreviation for thank you. But **i** has already been deciphered, and if we're going to start considering abbreviations then deciphering anything will become almost impossible.

- **aF at aD** - There are three words that start with the plain text letter **a**. These are **aF**, **at** and **aD**. The plain text words that start with **a** are **at, an**, **am**, and **as**. We've already deciphered **at**, and we just determined that **L** deciphers to **s**, which leaves **n** and **m** as possibilities for **F** and **D**. Looking at other words that contain **F** and **D** we find **atteDHt**. Plugging in **n** and **m** for **D** we get the following possible plain text words:

  atte**n**_t
  atte**m**_t

  The only solution that possibly forms a word is atte**m**_t, which would be the word **attempt.** This means that cipher text **D** maps to plain text **m**. It also means that cipher text **H** maps to plain text **p**.

  This also means that **F** must decipher to **n**.

Plugging these new letters into the message results in:

i Yeel e**n**e**m**B**s** CeaQe**s**t ReYe**ns**e i**s** the Ce**s**t Call. i Cill YaQe a**n** attaEQ t**o** the ea**s**t at tC**o** a**m**, Cait **s**eXe**n** **m**i**n**Wte**s** the**n** **m**aQe a**n** all **o**Wt atte**mp**t **on** the Ce**s**t.

Or

i _eel e**n**e**m**_**s** _ea_e**s**t _e_e**n**se i**s** the _e**s**t _all. i _ill _a_e a**n** atta_ _ t**o** the ea**s**t at t_**o** a**m**, _ait **s**e_e**n** **m**i**n**_te**s** the**n** **m**a_e a**n** all **o**_t atte**mp**t **on** the _e**s**t.

5. **Three character words – Part 2**. The first time through the three letter words we delayed **tCA** and **AWt** until we had more to work with. Now that we know that **A** deciphers to **o** the three letter words have become **tCo** and **oWt**.

   - **tCo** – This could decipher to **tao**, **too** or **two**. We can eliminate **tao** and **too** since **a** and **o** have already been deciphered. This means that it must be **two** and **C** must decipher to **w**.

   - **oWt** – This could decipher to **oat**, **oft**, **opt**, **ort**, **out** or apparently **oot** if you're in Scotland. The letters **a**, **p** and **o** can be eliminated as they've already been deciphered. The remaining letters **f r** and **u** can be tested using **minWtes** which also contains **W**. Since the only option that creates a word is **minutes**, this shows the solution is **out** which means **W** deciphers to **u**.

   Plugging **w** and **u** into the message results in:

i Yeel enemBs **w**eaQest ReYense is the **w**est **w**all. i **w**ill YaQe an attaEQ to the east at t**w**o am, **w**ait seXen min**u**tes then maQe an all o**u**t attempt on the **w**est.

Or

i _eel enem_s **w**ea_est _e_ense is the **w**est **w**all. i **w**ill _a_e an atta_ _ to the east at t**w**o am, **w**ait se_en min**u**tes then ma_e an all o**u**t attempt on the **w**est.

4. **Other clues**. At this point enough letters have been deciphered that you can look at the remaining cipher text letters that occur in more than one word.

   - **Q** - **Q** occurs in several places including **weaQest**, **YaQe**, **attaEQ**, and **maQe**. The only remaining letter that works in all the words is **k**.

   - **Y** - **Y** occurs in the words **Yeel**, **ReYense**, and **YaQe**. The only remaining letter that works in all the words is **f**.

   Plugging **k** and **f** into the message results in:

   i **f**eel enemBs wea**k**est Re**f**ense is the west wall. i will **f**a**k**e an attaE**k** to the east at two am, wait seXen minutes then ma**k**e an all out attempt on the west.

   Or

   i feel enem_s weakest _efense is the west wall. i will fake an atta_k to the east at two am, wait se_en minutes then make an all out attempt on the west.

5. **I will solve the puzzle Pat**. So far we've found the plain text characters **aefhiklmnpstuw** which leaves **bcdgjoqrvyz**. And the only letters left to decipher are **B**, **R**, **E** and **X**. These encrypted letters only occur once each, so if you don't immediately recognize the solution you can try plugging in the remaining plain text letters until you find one that creates a word. While it's theoretically possible to have multiple possible solutions, especially in short messages, the fact that we've already deciphered several letters makes it unlikely.

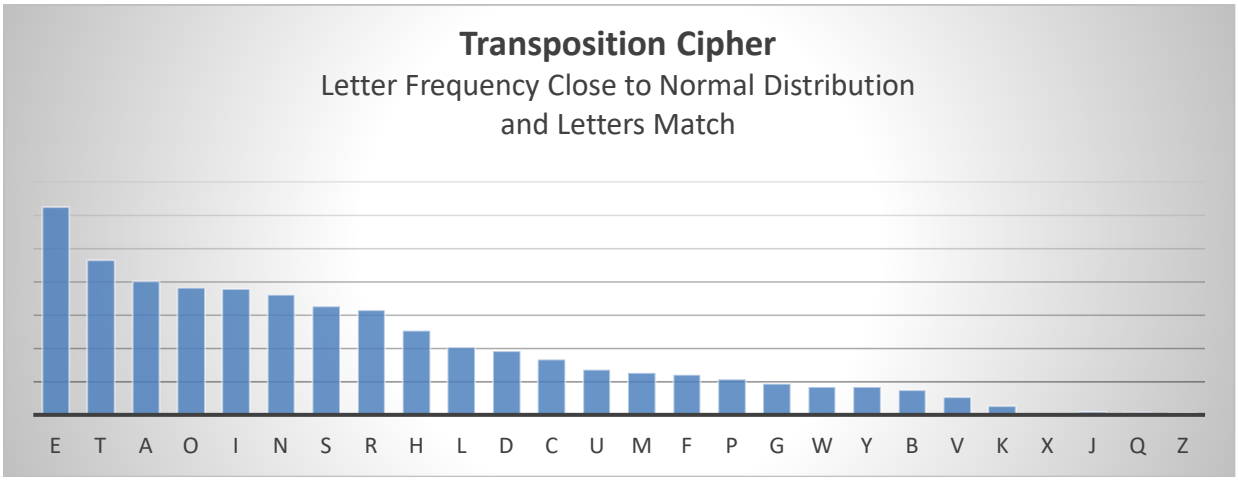
   Here's the decrypted message showing the solutions for **B**, **R**, **E** and **X**:

   i feel enem**y**s weakest **d**efense is the west wall. i will fake an atta**c**k to the east at two am, wait se**v**en minutes then make an all out attempt on the west.

   Notice that even though this message was relatively short at 199 characters, we were able to easily crack it using frequency analysis.

Frequency Analysis & Transposition Ciphers

Frequency Analysis is an excellent tool for cracking messages encrypted with substitution ciphers, but what about cracking transposition ciphers? Remember that transposition ciphers don't change the plain text characters to different characters, they just change their position. So, what would you expect to see if you perform a frequency analysis on a message that has been encrypted with a transposition cipher? That's right, the letter frequency should be close to the normal letter frequency. That is, there should be a lot of **e t o a i n** characters and very few **j q z**.



Even though frequency analysis won't help you decrypt transposed messages directly, it will provide a solid clue that the message has been encrypted with a transposition cipher. Armed with this knowledge, decrypting the message requires rearranging the letters to reveal the original message in a process known as anagramming. An anagram is word formed by rearranging the letters in another word. For example, the letters in **listen** can be rearranged to form the word **silent**. Anagrams can be entertaining, such as rearranging the letters in **The Morse Code** to **Here comes dots**, but that would only happen by coincidence with a transposition cipher.

With messages that have been encrypted with a transposition cipher anagramming consists of "unjumbling" letters from the cipher text. That is, the cipher text probably won't make real words, it will just appear as a bunch of scrambled letters. For example, the letters **kcclo** might be the cipher text. In this case you might be able to see that the plain text word was **clock**. But if not, you could try and brute force a solution, by either manually writing out every possible combination or by writing a program that writes every combination. Since there are 5 letters in **kcclo** there are only 5! or 120 possible ways the characters can be combined. A brute force attack might provide a solution for very short messages, but it's impractical for messages with more than a handful of characters. For example, a message with only 10 characters would have 10! or 3,628,800 different ways to combine the characters.

This leaves trying the different types of transposition ciphers such as the rail fence with different grid sizes or trying the route ciphers with different grid sizes and different routes.

Trying to brute force a solution by trying different ciphers and grid sizes might be alright if you are being paid by the hour and no one's safety depends on you cracking the message. But if time is of the essence, luckily there are several web sites and programs that will help speed up

the process. At these sites you feed in the cipher text, and then you can try different ciphers and grid sizes and view the results.

It's not expected that you become an expert at cracking transposition ciphers. But you should take away that you can use frequency analysis to determine if a message has been encrypted with a transposition cipher or a substitution cipher. If the letter frequency in the cipher text matches the normal text distribution for the same letters, then it's almost certain the message was encrypted with a transposition cipher. If the letter frequency in the cipher text follows the curve for normal text but the letters are different, then it's almost certain the message was encrypted with a substitution cipher.

Building Better Ciphers – The Cryptographers Fight Back

During the battle between cryptographers and cryptanalysts, frequency analysis really tipped the scales towards the cryptanalysts. There was no longer any assurance that a message encrypted with a simple substitution or transposition cipher would remain private. To fight back cryptographers came up with several schemes. In this section you'll learn about two of the simpler schemes, homophonic substitution, and the introduction of null characters.

Homophonic Substitution

The problem with all the substitution ciphers that you've learned about so far is that they do a 1-to-1 mapping between the plain text characters and the cipher text characters. This means that the frequently used characters like **e** will be mapped to a single cipher text character. Say for example that plain text **e** is mapped to the cipher text character **b**. If there are 100 **e** characters in the plain text, there will be 100 **b** characters in the cipher text. And given enough cipher text, frequency analysis can be used to make educated guesses about many of the characters.

But what if we mapped high frequency letters like **e** to more than one cipher text characters. That is, what if we mapped half of the plain text **e** characters to **b** and the other half to the character **4**? Can you see how this would change the cipher text, and how it would subsequently impact any frequency analysis run on the cipher text? That's right, it will throw the frequency analysis off by lowering the values for the characters **b** and **4**.

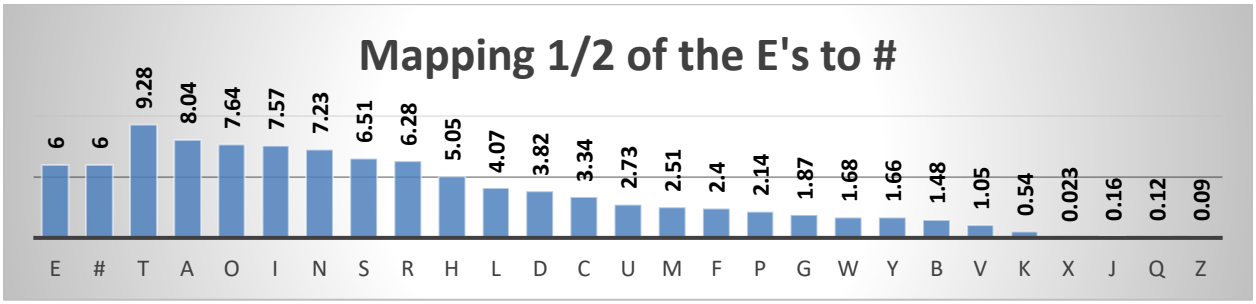
And what if you mapped the plain text **e** to even more cipher text characters? Since **e** occurs roughly 12 times more frequently than the average character, what would happen if we mapped **e** to 12 different cipher text characters? Any frequency analysis run on this type of cipher text would show 12 different characters that occur at the same rate as most other characters, effectively hiding the fact that all these 12 characters are plain text **e** characters.

This method of mapping high frequency plain text characters to multiple cipher text characters is called *homophonic substitution* and it's first known use was in 1401 by Francesco I Gonzaga, the Duke of Mantua, in what is now northern Italy. It's called homophonic because one of the definitions of a homophone is two or more symbols that denote the same sound. (The other

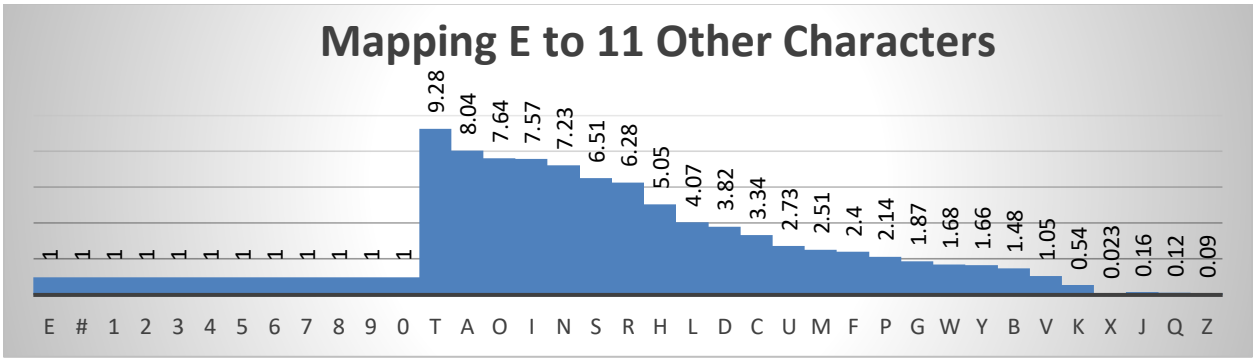
definition is two or more words that sound the same but mean different things like to, two and too.)

You might see one glaring issue with homophonic ciphers, which is that they require additional characters for the cipher text. If you start with 26 plain text characters and want to map the high frequency letters to multiple cipher text characters, you're going to need more than 26 cipher text characters. There are an infinite number of ways to do this, for example adding numbers to the cipher text, or using both upper case and lower case characters in the cipher text, or using characters from a different alphabet, or even making up special symbols for the cipher text.

The exact number of extra cipher text characters needed depends on how flat you want to make the frequency analysis curve. For example, frequency analysis on any large section of plain text shows that normally the letter **e** occurs about 12 times more often than the average. The **e** portion of the graph can be lowered by mapping half of the **e** characters to a different character, say for example #. In this case, each of these two characters **e** and # will occur about 6 time more often than the average. So, a frequency analysis graph will still show them as high spots, but only about 6 times higher than the average.



If you use 3 text characters for **e** the resulting frequencies for these 3 characters will be about 4 times higher than average on a frequency analysis graph. Each time you map **e** to an additional cipher text character it will reduce the number of occurrences for each of the resulting cipher text characters and lower the corresponding bar on the graph. To drastically flatten the graph bars for **e** you can map it to 12 different text characters. These 12 cipher text characters will each have a frequency of ~1 on analysis graph, which would be very effective in "hiding" **e**.



If you repeated this process in mapping all the characters you could essentially flatten the frequency curve. The letters **k x j q** and **z** will always be lower than 1%, so their portion of the curve will show a slight dip. But the frequency curve would be so flat it would stop cryptanalysts

from using frequency analysis as a tool. The cost, if you do this for all the characters that occur more than 1% of the time, is you will end up needing roughly 100 cipher text characters. Any number that normally occurs more than 1% of the time will need at least two cipher text characters. This means your encryption and decryption tables will be a little longer and more complicated, but that seems like small price to pay for the extra security.

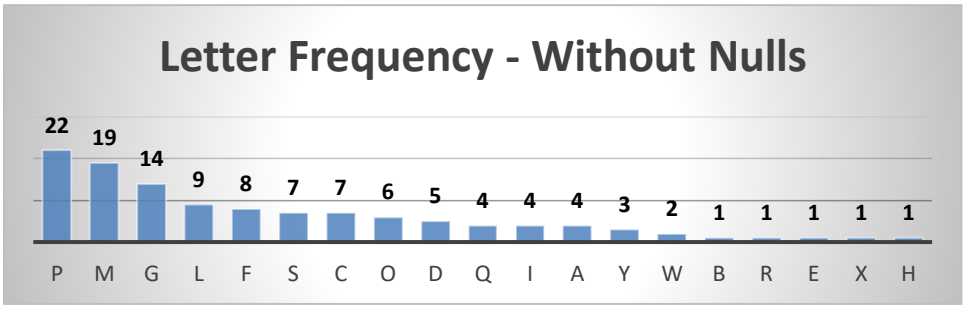
Adding Nulls

Another simple but devious way of impeding frequency analysis is to add null characters to the cipher text. A null character is a character that the recipient knows means nothing and should just be thrown out. For example, say that you map all the plain text characters to upper-case alphabet characters using a normal simple substitution cipher. Before you send the message you also randomly insert lower-case letters that mean nothing. If you add in enough lower-case letters it will throw off the frequency analysis and make deciphering the message much more difficult. It won't make it any harder for the intended recipient to read the message. They'll know to remove the upper-case letters before they decipher the message, so they'll have no trouble reading the message.

For an example, let's start with the same encrypted message we used in the previous demonstrations. The original encrypted message is:

> O YPPS PFPDBL CPGQPLM RPYPFLP OL MIP CPLM CGSS. O COSS YGQP GF GMMGEQ MA MIP PGLM GM MCA GD, CGOM LPXPF DOFWMPL MIPF DGQP GF GSS AWM GMMPDHM AF MIP CPLM.

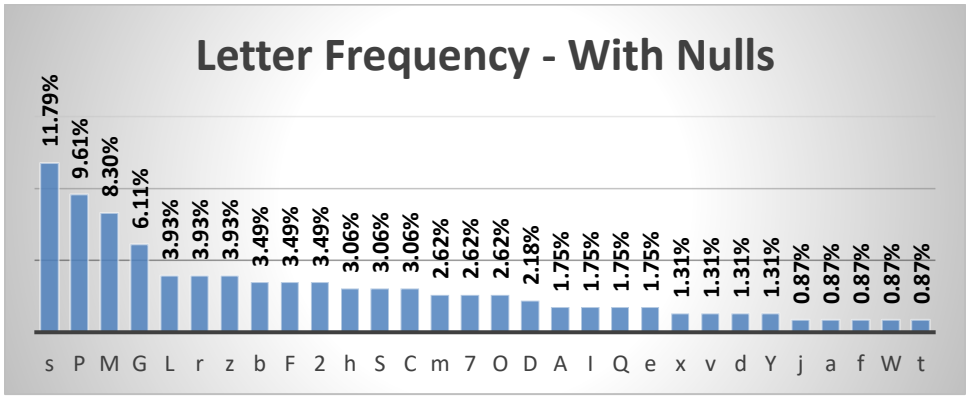
Running a frequency analysis on this produces the following results:



Next, the message is modified by adding several lower-case letters and numbers. This results in the following cipher text:

> 7 srO YvsdePPfS P4Fdd2PDvBL res CPvGfQPLM bhj RsPYPFLP 2sm OLa MaIP zx CbPLM zsxs CGSSs2. srO COSS mzz YGQ9P trGF jbs5 GsMMGEQ MAe eMIP slp0 hs7s 1PGLMt bhs rGM sr MCsA GD7s, CsGO2M 2sr LPXPF bhs DOFWMPL MIhPF DGQP mzz GFs2 bhs GSS sr AW2M 7sn GMMPDHM AFm 7MIP CmPLM s2c. r7b zx mzz bhs

Running a frequency analysis on the modified cipher text shows how this can make it more difficult to identify the high frequency characters and make the task of deciphering more difficult for anyone who doesn't remove the null characters. Notice that in this case the letter **s** occurs most frequently, which would deceive the cryptanalyst into thinking that it maps to **e**.



Using homophonic ciphers and adding null characters were just two ways that cryptographers tried to strengthen their schemes, mainly to defeat frequency analysis attacks. I'm sure if you thought about it for a few minutes you could come up with other ways. However, when it comes to understanding modern ciphers it's not critical that you memorize every step in the evolution of ciphers. What's important is that you understand how substitution ciphers work and how they can be broken with frequency analysis.