

Tiny Book of Simple Cryptography



Alfred C Thompson II

Cover image created by Microsoft CoPilot

[Tiny Book of Simple Cryptography](#) by [Alfred Thompson](#) is licensed under CC BY-NC-SA 4.0 To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-sa/4.0>

Contents

Codes or Ciphers?	1
Introduction	2
Caesar Cipher	2
Introduction	2
Encrypting	2
Decrypting	3
Cryptography Issues	3
Project Suggestions	4
Vigenère cipher	5
Introduction	5
Encrypting	5
Decrypting	5
Cryptography Issues	5
Project Suggestions	6
Wheel Cipher	7
Introduction	7
Encrypting	7
Decrypting	7
Cryptography Issues	7
Project Suggestions	7
Notes of Interest	7
One Time Pad	8
Introduction	8
Encrypting	8
Decrypting	8
Cryptography Issues	8
Project Suggestions	8
Polybius Square	10
Introduction	10
Encrypting	11
Decrypting	12

Cryptography Issues.....	12
Project Suggestions.....	12
PigPen Cipher	13
Introduction	13
Encrypting	13
Decrypting.....	14
Cryptography Issues.....	14
Project Suggestions.....	14
Columnar Transposition Cipher	15
Introduction	15
Encrypting	15
Decrypting.....	16
Cryptography Issues.....	16
Project Suggestions.....	16
Keyword Columnar Transposition Cipher	17
Introduction	17
Encrypting	17
Decrypting.....	18
Cryptography Issues.....	18
Project Suggestions.....	18
Random Block Transposition Cipher	19
Introduction	19
Encrypting	19
Decrypting.....	19
Cryptography Issues.....	19
Project Suggestions.....	19
Steganography	20
Encrypting	20
Introduction	20
Decrypting.....	20
Cryptography Issues.....	20
Project Suggestions.....	21
Bacon's Cipher	22

Introduction	22
Encrypting	22
Decrypting	23
Cryptography Issues	23
Project Suggestions	23
Book Ciphers	25
Introduction	25
Encrypting	25
Decrypting	26
Cryptography Issues	26
Project Suggestions	26
Playfair cipher	27
Introduction	27
Encrypting	27
Decrypting	28
Cryptography Issues	28
Project Suggestions	28
Useful Links	29
Code Samples	30
Introduction	30
Padding a string with random letters	30
Creating an array of numbers in a random order	30
Determining the ordinal position of a letter	31
Building a String Representation of a Binary ASCII value	31
Determine if a letter is uppercase or lowercase	31
Appendix	32
Resource Links	32
Frequency of Letters in English	32
Non-English Alphabets	33

Codes or Ciphers?¹

Before we get too far into things, one common question is “what is the difference between a code and a cipher?” Most people think of them as the same, but they are different. A code involves replacing one word with another word or symbol. A cipher involves an algorithm that transforms letters or groups of letters according to an agreed set of steps. A code requires a code book that explains what words are substitutes for what other words. Ciphers do not require a code book although in some cases there is a shared secret that helps to start the algorithm. This book is primarily about ciphers.

¹ <https://www.khanacademy.org/computing/computer-science/cryptography/ciphers/a/ciphers-vs-codes>

Introduction

Modern cryptography is filled with complicated algorithms, intricate mathematics, and sophisticated software. It was not always the case. Before the age of computers many, simpler, encryption methods were in use. In this booklet, we will look at some older methods and explore writing code to encrypt and decrypt messages.

Some of these are known as Substitution ciphers² which means that letters, individually or as a group, are replaced with other letters. There are substitution ciphers where letters are replaced with other symbols or numbers as well. For example, ASCII³ is a cipher that replaces letters and other characters with numbers.

Other simple ciphers are transposition ciphers⁴. In transposition ciphers letters are moved around according to a pre-determined pattern.

Throughout this booklet are footnotes that link to articles on Wikipedia that provide more information about the topics under discussion. These may be used as a starting point for more research and more details.

This booklet is intended as a starting point for experimentation and further learning. It is not intended as a definitive work.

Caesar Cipher⁵

Introduction

The Caesar cipher is one of the oldest and most well-known ciphers. While it may or may not have been invented by Caesar it became famous as something he used frequently. The cipher is simple in that each letter is replaced by a letter some number of positions after it in the alphabet. Caesar rotated the letters by 3 so that today we would replace the letter A with the letter D, the letter B with E, and so on.

The number used to encrypt a message is a shared secret. A shared secret is something that the sender and received both know but that no one else knows. Shared secrets must be shared before messages can be sent. This is a weakness of many ciphers.

Encrypting

Encrypting a message becomes a matter of knowing the ordinal position of a letter, adding the offset, and replacing the original letter with the letter at the indicated position. What if the position indicated is beyond the count of letters in the alphabet? In that case we start over at the beginning. Modulus arithmetic can help us there. The modulus operation returns the remainder of a division operation. For example, if we add 13 to position 16 (T) we get 29 which is beyond the number of letters in the English alphabet. 29 modulus 26 (the number of letters in our alphabet) returns 3 which leads us to D (A would be 0).

² https://en.wikipedia.org/wiki/Substitution_cipher

³ <https://en.wikipedia.org/wiki/ASCII>

⁴ https://en.wikipedia.org/wiki/Transposition_cipher

⁵ https://en.wikipedia.org/wiki/Caesar_cipher

The formula is $\text{Location} = (\text{Position} + \text{Offset}) \% 26$

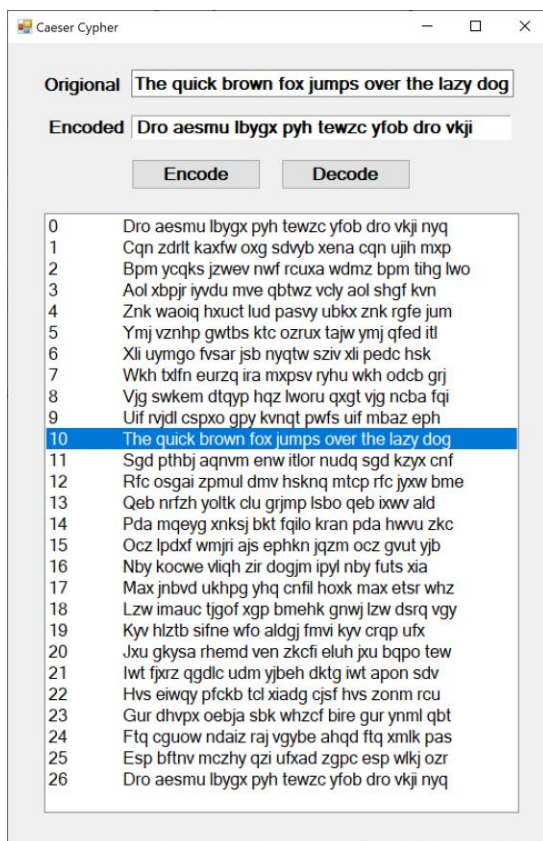
Decrypting

Decrypting an encrypted message is done by subtracting the offset from the encrypted value to get the location of the original letter. Our problem here is that our location may be before the beginning of the alphabet (less than zero). This means that we must check for the case of a negative location and use that value to subtract from 26 to get the correct location.

Cryptography Issues

There are several issues with this cipher that keep it from being secure in the modern world. The first problem, which it shares with many substitution ciphers, is that it is susceptible to breaking through letter frequency analysis. Some letters occur more frequently in writing than others. In English, E is the most used letter. The letter Z is used least frequently (there is a table showing frequency in the appendix). We can count the number of occurrences of each letter in an encrypted message. Given a large enough sample we can make some good assumptions about what letter is replacing what other letter. With the Caesar cipher we can deduce the rotation based on the letters we believe we have identified and so decrypt the rest of the message.

Modern computers give us our second easy way to break this encryption. There are only 25 possible rotations. Zero and 26 are the same as not encrypting at all. A simple program that implements all 25 possibilities will quickly show a clear plaintext result. The following image shows one implementation. We can easily see that the encrypting here used a rotation of 10 characters.



The fact that secrets must be shared and that someone may discover the secret is also an issue as it is with any cipher that requires shared secrets.

Project Suggestions

1. Write a program to encrypt and decrypt using the Caesar cipher.
 - a. Include options to enter either plaintext or encrypted messages
 - b. Include an option to set the number of characters to rotate letters
2. Write a program to count the letters in the text in a data file. Compare those counts with the chart in the appendix.

Vigenère cipher⁶

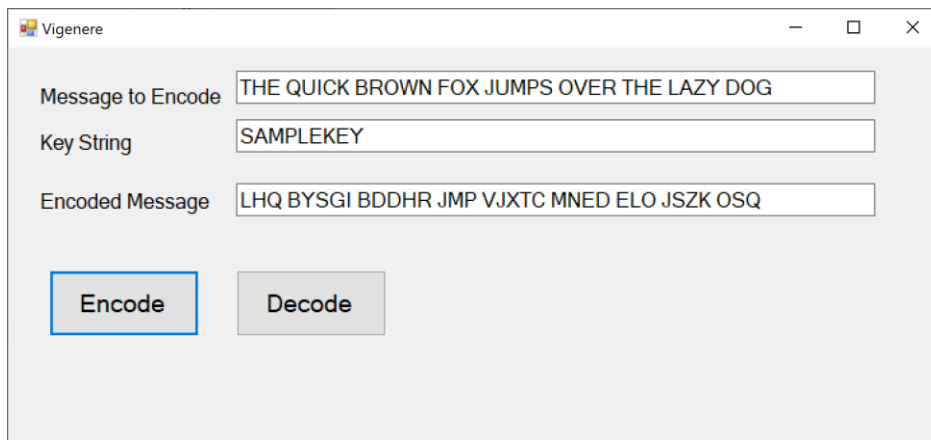
Introduction

As we have seen, the Caesar cipher is both easy to use and easy to break. The Vigenère cipher takes the Caesar cipher a big step further and was considered unbreakable for hundreds of years. This cipher rotates each letter by a different number of places. The rotation values are determined by a key, normally a phrase, that both the sender and receiver agree to in advance. Another shared secret.

Encrypting

The ordinal position for each letter is used as the rotation factor for the corresponding letter in the plain text. For example, in the figure below, S or 18 is the offset for the letter T in the plaintext which gives us the letter L. The letter A, the second letter in the key string, is in position zero so the second letter in the plaintext, H, is not changed at all. This is not considered a problem as anyone intercepting the message would assume that the letter had been changed.

If the key phrase has fewer letters than the plaintext being encrypted, the phrase is used repeatedly until the whole message is encrypted. Longer keys create messages that are harder to break.



The screenshot shows a window titled "Vigenere" with three text input fields and two buttons. The first field, labeled "Message to Encode", contains the text "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG". The second field, labeled "Key String", contains "SAMPLEKEY". The third field, labeled "Encoded Message", contains "LHQ BYSGI BDDHR JMP VJXTC MNED ELO JSZK OSQ". Below the fields are two buttons: "Encode" (highlighted with a blue border) and "Decode".

Decrypting

The key phrase is used to decrypt the message once it has been received. Therefore, both sender and receiver must know the key.

Cryptography Issues

These messages are usually written with all letters in either upper or lower case. Having mixed case letters gives information about which letters may indicate the start of words such as the first word in a sentence or the start of a proper noun. Spaces also give away information about the size and number of words in a message. Encrypted messages are more secure if spaces are removed from the message.

Keys must be shared in advance or transmitted in some other secure manner. This can complicate the use of the code because it makes it hard to use the code with new senders or receivers.

⁶ https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

Modern super computers can break this cipher though several proven attacks. It will probably be safe from casual readers but not from a major organization or government.

Project Suggestions

1. Write a program to encrypt and decrypt messages using the Vigenère cipher.

Wheel Cipher

Introduction

The Wheel Cipher⁷, sometimes called a Jefferson disk ⁸or a Bazeris cylinder, was originally invented by Thomas Jefferson in 1795. There is some question about it ever being made or used at that time. It was independently invented in 1891 by Étienne Bazeries in France.



The wheel cipher consists of a number of disks with the alphabet in different random orders. Jefferson's cylinder had 36 wooden wheels on an iron rod. Each disk had the whole alphabet in a different random order. Users would need identical cylinders with the disks in the same order.

Encrypting

Messages were encoded by lining up the letters of the plain text at the top of the cylinder. A row of letters elsewhere on the disk made up the encoded message. The letters in the selected line would be transmitted.

Decrypting

The receiver of the encoded message would line up their own, identical cylinder so that letters matched the message. The receiver would look to a specific line or search the cylinder for a row that made sense. That would be the plaintext message.

Cryptography Issues

This was a solid scheme for its time but not secure today. It is susceptible to several code breaking methods. If the same order of the disks is used repeatedly frequency analysis can be used to break the code. Modern computers make brute force methods, checking all possible combinations, can be effective.

For good security the order of disks would be changed regularly. Of course knowledge of changes have to be shared without a potential attacker being aware of them.

Project Suggestions

1. Write a program to encode and decode using a Wheel Cipher. Note: Both encryption and decryption must use the same virtual disks.

Notes of Interest

Image above from Wikimedia

https://commons.wikimedia.org/wiki/File:Jefferson%27s_disk_cipher.jpg

The gift shops at both Monticello and Mount Vernon sell replicas of the Jefferson disk.

- <https://www.monticelloshop.org/monticello-secret-decoder-wheels/>
- https://shops.mountvernon.org/products/mount-vernon-cypher-wheel-secr-17621?_pos=1&_psq=wheel&_ss=e&_v=1.0

⁷ https://en.wikipedia.org/wiki/Jefferson_disk

⁸ <https://www.monticello.org/research-education/thomas-jefferson-encyclopedia/wheel-cipher/>

One Time Pad

Introduction

The one-time pad⁹ is generally considered the unbreakable code. At least within some given constraints discussed below in Cryptography Issues. The concept is that the message is encrypted using a set of random numbers with the set being the same length as the plain text message. A set of random numbers is used only once which is what gives this cypher its name.

Encrypting

Encrypting using a one-time pad is very similar to encoding using a Vigenère cipher except that the rotations are determined by the numbers in the one-time pad and not from a keyword.

Decrypting

Decrypting a crypto text that was encoded using the one-time pad requires that the same one-time pad that was used to encode the message is used to determine the rotation of letters to recover the plaintext.

Cryptography Issues

This cipher is useful only if both the sender and receiver have the same set of numbers or pad. This means that some way of passing this information along securely is possible. If a third party were to get a copy of the pad the code is insecure and can not be used. Also, the set of numbers has to be truly random. This is a difficult problem¹⁰. Computer generated lists of random numbers are not truly random. They are technically referred to as pseudo random. They are random enough for many uses but not for cryptography or other applications like lotteries where true random numbers are necessary.

There are several websites on the internet that promise to provide truly random sets of numbers and two of them are listed here.

- <https://www.random.org/integers/>
- <https://numbergenerator.org/true-random-numbers>

Trusting internet websites for random numbers for cryptography is a risky venture and companies and governments who depend on security often have their own sources of random numbers. These websites provide lists that can be used for testing program code and for casual use. There are ways of verifying the randomness of numbers that can be used.¹¹ That is beyond the scope of this book.

Project Suggestions

Create a program that encodes a plain text message using a list of random number read from a file. Strip spaces from the plaintext before encoding. The program must read at least as many numbers as the plaintext is long. The program must also be able to decode a message using the same list of random numbers.

Option: Most pseudo random number routines will generate a list of numbers based on a seed value. Calling the routine twice with the same seed will result in duplicate lists. This is a possible way of

⁹ https://en.wikipedia.org/wiki/One-time_pad

¹⁰ https://en.wikipedia.org/wiki/Random_number_generation

¹¹ https://en.wikipedia.org/wiki/Random_number_generation#Post-processing_and_statistical_checks

creating a program if you don't have easy access to files with lists of random numbers. It is not a secure method and should not be used for serious cryptography.

Polybius Square

Introduction

The Polybius¹² Square¹³ was discovered by Greek scholars named Cleoxenus and Democleitus and made famous by the historian Polybius. It was used to send messages more easily. The basic idea is a 5 by 5 grid holding the letters of the alphabet. The letters were identified by the x and y coordinate numbers. Polybius used the Greek alphabet with 24 letters which fit nicely in a 5 by 5 grid. A (alpha) was 11. Ω (Omega) was 54.

	1	2	3	4	5
1	A	B	Γ	Δ	E
2	Z	H	Θ	I	K
3	Λ	M	N	Ξ	O
4	Π	P	Σ	T	Υ
5	Φ	X	Ψ	Ω	

The English alphabet has 26 letters which doesn't fit so the same box is often used for the letters "I" and "J". Each letter is then represented by a two-digit number that identifies its location in the square.

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

Alternatives to "I" and "J" are the letters "C" and "K" sharing a box, but other pairs can be used as well by prior agreement.

¹² https://en.wikipedia.org/wiki/Polybius_square

¹³ <https://crypto.interactive-maths.com/polybius-square.html>

In some cases, a 6 by 6 square that includes the digits zero through nine is used. This also avoids the need for two letters to share a box as well as avoiding the need to spell out numbers.

	1	2	3	4	5	6
1	A	B	C	D	E	F
2	G	H	I	J	K	L
3	M	N	O	P	Q	R
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

Six by six grids are also used for alphabets that have more than 25 or 26 letters. Russian has 33 letters for example.

In the case of the 6 by 6 English square the letter “A” is identified by 11. The number “1” is identified by 54. The letter “O” is identified by the number 33.

Having the square in a standard order is easy and was the originally way squares were used but to hide a message we want something different.¹⁴ Using a keyword and ignoring any duplicate letters is one possible way and probably the most common. Other potential ways to build a square include reversing the alphabet or sharing a randomly generated square. The figure below uses the keyword “Polybius” as the beginning of the square with the rest of the alphabet following without duplicating any letters from the keyword. As shown in the next figure.

	1	2	3	4	5	6
1	P	O	L	Y	B	I
2	U	S	A	C	D	E
3	F	G	H	J	K	M
4	N	Q	R	T	V	W
5	X	Z	0	1	2	3
6	4	5	6	7	8	9

Encrypting

The first step in encrypting a message is to build the Polybius Square. The letters of the keyword are entered in to grid first followed by the rest of the alphabet, being sure not to duplicate letters. Each

¹⁴ <https://en.wikipedia.org/wiki/Polybius#Cryptography>

letter in the plaintext is located in the square and the x and y coordinates of that location are used to represent the letter.

Using the square above the plaintext "SAMPLETEXT" becomes "11215200205133510433".

Decrypting

Decrypting also starts with creating a square. The encoded message is broken into two-digit codes which are further broken down to x and y coordinates and used to identify the original letter from the plain text.

Cryptography Issues

Like any other single transposition, the Polybius Square is susceptible to frequency analysis. Modern computers can also break it via brute force methods of trying all possible combinations. This encryption has very limited use if real security is needed.

Project Suggestions

Creating a program to create and use a Polybius Square is a good project. A simple 5 by 5 square without a keyword is the easiest first project.

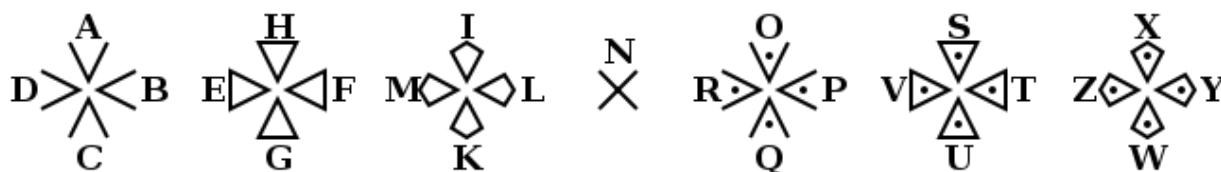
Creating a Polybius Square using a keyword is more complicated than not using a keyword in the square. This makes a good second project.

PigPen Cipher¹⁵

Introduction

One of the simplest and oldest methods of encoding messages is by replacing letters with other symbols. After all, letters are really symbols for sounds. The PigPen cipher is one well-known example. It has been called several other names over the years including the masonic cipher, Freemason's cipher, Napoleon cipher, Templar Cipher, and tic-tac-toe cipher. Versions of it have been used at least as early as the 16th century.

For example, the Knights Templar used a version based on the Templar Cross.



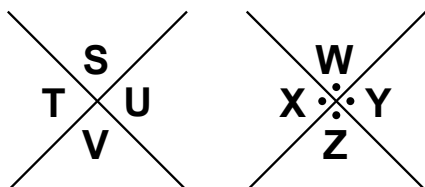
Templar Cipher

Other versions are based on grids or X shapes with additional items like dots.

Encrypting

The figure below shows one popular version of the PigPen cipher. Each section of the grids defines a letter with a dot in the section differentiating the different grid.

A	B	C	J	K	L
D	E	F	M	N	O
G	H	I	P	Q	R



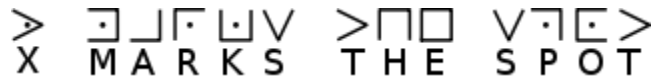
PigPen Grid

The message is encoded by using the grid segment, empty of letters but including a dot if necessary, to represent the letters that make up the message. For example, the letter A would be represented by the lines for the top left of the first grid – A and B would be the lines for the top center of the grid - B. The letter W would be the lines and a dot for the top section of the last X grid – W.

¹⁵ https://en.wikipedia.org/wiki/Pigpen_cipher

The alphabet looks like this: ABCDERGHIJKLMNOPQRSTUVWXYZ

A complete message would consist of the various symbols as displayed in the image below.



Coded Message with Decoded Result

Messages are encoded by referring to the diagrams, but some users may memorize the code. The memorization is made easy by the general organization of the grids.

Decrypting

Decrypting a message is a matter of comparing the symbols to the encoding grid. Each symbol can be compared to the grid and finding the corresponding part of the grid and reading the letter in that location.

Cryptography Issues

The use of simple grids makes it easy to share the code as well as to encrypt and decrypt messages. It is easy to reproduce a key for encrypting or decrypting messages if one knows the pattern used. For this reason, many different patterns can be used. For example, grid X grid X would give a different result than grid XX as was used in our example above. Other variations are moving the letter in a grid down first and then across or using only grids with different numbers of dots in sections. Of course, the pigpen method could be combined with other forms of substitution such as Caesar or Vigenère. In any case, the receiver must know the patterns used in advance. If the patterns are not changed from time to time the cipher becomes easier to decode (break).

Project Suggestions

There is a TrueType font of the PigPen symbols available at https://fontstruct.com/fontstructions/show/447940/cipher_code. It can be used to display encoded messages. For example, the word “pigpen” could be displayed as pigpen. This font was used to display encoded messages in this document.

Create your own pattern for encrypting messages. Use that pattern to send a message to someone else and see if they can decrypt it.

Columnar Transposition Cipher

Introduction

In a columnar transposition, plaintext letters are written out in rows and the encrypted message is built by taking the letters from columns. The simplest form of this is to take the columns in order but more secure ciphers process the columns in an order specified by a keyword or previously shared secret.

Encrypting

In this simple example (figure below) we are writing the message TESTMESSAGE in a four-by-four grid. We are padding trailing spaces with the letter Z. In some variations, the end of the message is padded with randomly selected letters. In other variations, no padding is used.

T	E	S	T
M	E	S	S
A	G	E	E

The coded message would be TMAEESSETSE if we move down from the first column to the last in order.

In the figure above I have written the message in lowercase but padded the string with uppercase letters so that you could see the padded letters. Ideally you would use the same case for the padding as you would use in the message itself to avoid giving a code breaker more information.

There are any number of ways of specifying the number of columns and the order in which they are to be used to build the encrypted message. For example, a key word can be used where the number of letters in the word indicates the number of columns, and the alphabetical order of the letters indicates

the order in which to process the columns. The number of columns and order could be shared in advance as well.

Decrypting

Decrypting is the opposite process. One writes into the columns and reads across the rows. Two dimensional arrays are ideal for programming this process. There are other ways of doing it and I leave that to your imagination.

Cryptography Issues

This method of encrypting messages was in common use until the 1950s when computers made it possible to try many combinations of columns in a relatively short period of time. As with other methods, spaces and mixed case letters provide a potential code breaker with useful information. If the message is prefixed with a key word that indicates the number of columns and the order in which to process them breaking one instance of a code will often give away future instances that use different key words.

Project Suggestions

1. Write a program to rearrange the letters in a message using a specified block size (number of columns).
2. Write a program to rearrange the letters in a message using a number of columns and an order to process them based on a key word.

Keyword Columnar Transposition Cipher

Introduction

A Keyword Columnar Transposition Cipher¹⁶ is a columnar transposition that moves columns based on a shared key word or key phrase. Plaintext is written out under columns identified by a keyword or phrase. The length of the keyword is the number of columns. An initial square with the key phrase “amystery” might look like the figure below.

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

The next step is to sort the columns based on the alphabetically sorted letters in the keyword. For example, the “t” in “amystery” indicates the 5th column (position 4 when counting from zero) but is the sixth letter in the word alphabetically. That means that the “t” column in the plaintext square becomes the sixth (5th position) in the encrypted square.

Note that squares that are not filled by letters from the plaintext may be either left as empty spaces, as here, or filled with random letters.

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

The resulting grid can be read out either by rows or columns. Reading out by rows, the plaintext “thequickbrownfoxjumpsoverthelazydog” using the keyword “amystery” would be encoded as “tihcquekbrownoxjouvpmsmeratzelhy d o g”. Both the sender and the receiver have to know the keyword and to read by rows or columns.

Encrypting

The first step in encoding a message is to remove the spaces, as is typical for coded messages, and to build the grid. The message is written out row by row. The next step is to determine the alphabetical

¹⁶ https://en.wikipedia.org/wiki/Transposition_cipher#Columnar_transposition

order of the letters in the keyword/phrase. Then the columns are reordered based on the order of the keyword letters. The message in the block is listed in order by row or column.

If programming, the most difficult part of the process may be tracking the order of the columns. There are probably many possible ways to do this. One way is to build an array of two-character symbols, the first character a letter from the keyword and the second an integer indicating the original location of the letter. This array can be sorted and the position in the array indicating the destination column and the number indicating the source column.

Decrypting

Decoding is much the same as the encryption process. The grid is made using the encoded message. The original and sorted order of the letters is used to reorganize the columns back to their unencoded positions.

Cryptography Issues

This cipher can be broken by guessing at the number of columns, building squares, and looking for anagrams. This method also lends itself to brute force methods that are practical with powerful computers and good software.

Project Suggestions

Create a program that encodes and decodes a message using a keyword and associated columnar transposition.

For more advanced work, allow a choice between outputting the cipher text by row or by column and the user's option.

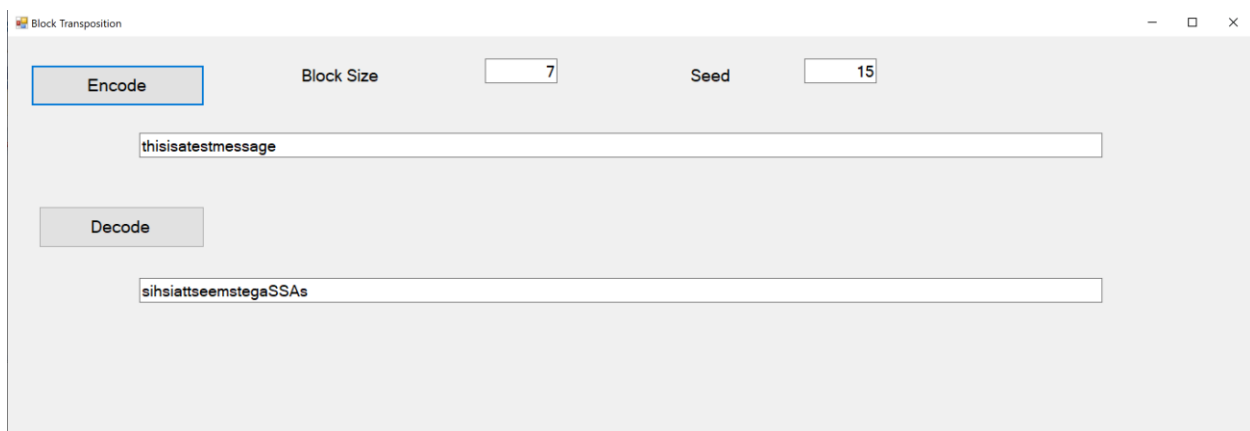
Random Block Transposition Cipher

Introduction

Random number generators as used in most programs are not truly random. They are known as pseudo random numbers¹⁷. While this has disadvantages for most cryptographic usage. They can be used for a simple block transposition cipher. That is because if the program provides the random number routine with a specified seed value the random number generator will calculate the same not quite random values.

Encrypting

For this method a block size is selected. The letters in each block are rearranged in a random order. The random order is determined by a specific pseudo random number generated with a specific seed. The seed and the block size are shared secrets. As with other transposition ciphers, the plaintext is padded with random letters to avoid spaces and other information that would provide extra information to potential cipher breakers.



The screenshot shows a window titled "Block Transposition" with a light gray background. At the top, there are two input fields: "Block Size" with the value "7" and "Seed" with the value "15". Below these are two buttons: "Encode" (highlighted with a blue border) and "Decode". Under the "Encode" button is a text input field containing the plaintext "thisisatestmessage". Under the "Decode" button is a text input field containing the ciphertext "sihsiattseemstegaSSAs". The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

Decrypting

Decrypting involves using the same seed, shared secret, and the same pseudo random number generator to build the same order of transpositions in the same size block to move letters to their original position.

Cryptography Issues

Block sizes that are too small and which may be tried give too few possibilities for rearrangement. Larger block sizes are harder to crack as there are more possibilities. Brute force methods of cracking are still possible with fast computers.

Project Suggestions

Write a program that accepts a message with a block size and random number generator seed and encrypts and decrypts as message

¹⁷ https://en.wikipedia.org/wiki/Pseudorandom_number_generator

Steganography

Steganography “is the practice of concealing a file, message, image, or video within another file, message, image, or video.”¹⁸ Steganography hides the fact that there is a message. A file can be passed around in various forms without anyone, in theory at least, knowing that information is being passed along with it.

Encrypting

Introduction

There are many ways of hiding messages in an image. One way involves using the way that colors are stored in images. Colors are stored in a combination of red, green, and blue. Each color, in the most common form includes 255 levels of each of the three primary colors. The difference between 250 and 251 for one color is generally far too subtle for the casual viewer. We can take advantage of this to store information using one bit in each pixel. For example, we can use the last bit of the color blue and eight pixels to represent one ASCII character.

Our encoding routine would select a known location in the image to start the message. The first 8 pixels might be used to represent the total number of characters in the message.

For each character, including the letter count, our program would build a pattern of ones and zeros. The least bit in eight pixels would be set or left alone to match the pattern. Many bits would not be changed because they were already set the way our message requires them to be. The fact that the decryption program knows that the pixel is part of the message is all that is needed for this to work. Unchanged pixels make detection more difficult.

One thing to make note of is that many forms of compression will make changes to the file that make it almost impossible to recreate the hidden message. Image formats that use lossy compression¹⁹ to save space will generally destroy the hidden message. Image formats like JPEG are examples of this. Using a JPEG image is not going to be reliable.

Decrypting

Decrypting involves first knowing the message in the file and where to look for it. Once the location and formatting of the message is known the decoding program checks the values that are stored and constructs a legible message.

Cryptography Issues

Comparing a file with a message with the original file, before a message was encoded in it, will provide many clues to the message hidden it.

¹⁸ <https://en.wikipedia.org/wiki/Steganography>

¹⁹ [Lossy compression - Wikipedia](#)

Project Suggestions

1. Create a program to hide a message in an image file (recommended BMP or PNG).
 - a. Program should modify the bits in known locations of an image file and rewrite the new modified file.
 - b. Program should read a file with an encoded message and decode it.

Bacon's Cipher

Introduction

Bacon's cipher²⁰ is a well-known and simple cipher. It's considered a form of [steganography](#)²¹. Unlike most steganography, the code is hidden in the text and not an image. The format is a 5-bit binary encoding where each letter of the plaintext is replaced by a group of five of the letters 'A' or 'B'. This binary code has A as 00000 or "aaaaa" and Z is 11010 or "bbaba". There are variations and the original version used the same codes for the letters "I" and "J" and for "U" and "V". The usual implementation is to use one type face for "a" and a different type face for "b".

Often this is done with a serif and a sans serif font. One option, and the first one I use below, uses italics and non-italics.

thequickbrownfoxesjumpedoverthelazy

One other option I have used is mixing upper-case and lower-case letters.

The QUick brOwn FoXes JUmPed ovER tHe lAZy

One semi-famous example of the use of serif and Sans Serif fonts is the Cipher on the William and Elizebeth Friedman tombstone.²² The biography of *Elizebeth Friedman, The Woman Who Smashed Codes: A True Story of Love, Spies, and the Unlikely Heroine Who Outwitted America's Enemies*²³, is a seriously recommended read.

There are also examples of using this encoding in pictures of groups of people. Some people facing straight ahead, and some turned to the side. Just about any possibility of two different options would work. It's a flexible system.

Encrypting

Encrypting a message requires a plaintext message where the secret message will be hidden. Each letter in the secret message must be mapped to its A/B code. The plaintext message is then modified so that each group of 5 letters is set to the appropriate mix of A or B indicators. Secret messages traditionally do not include spaces because spaces make it easier to break encoded messages into words. The plaintext may have spaces, but they will be left unchanged by the encryption process. Spaces will be ignored during decoding as well.

The following table shows the various encoding using both A/B code and Binary codes.

²⁰https://en.wikipedia.org/wiki/Bacon's_cipher

²¹ <https://en.wikipedia.org/wiki/Steganography>

²² <https://www.elonka.com/friedman/>

²³ <https://smile.amazon.com/Woman-Who-Smashed-Codes-Outwitted-ebook/dp/B01M0EOI6I/>

Letter	Code	Binary	Letter	Code	Binary
A	aaaaa	00000	N	abbab	01101
B	aaaab	00001	O	abbba	01110
C	aaaba	00010	P	abbbb	01111
D	aaabb	00011	Q	baaaa	10000
E	aabaa	00100	R	baaab	10001
F	aabab	00101	S	baaba	10010
G	aabba	00110	T	baabb	10011
H	aabbb	00111	U	babaa	10100
I	abaaa	01000	V	babab	10101
J	abaab	01001	W	babba	10110
K	ababa	01010	X	babbb	10111
L	ababb	01011	Y	bbaaa	11000
M	abbaa	01100	Z	bbaab	11001

Decrypting

Decrypting involves viewing the groups of five characters, determining the A/B or 0/1 values, and matching up with the appropriate letters. Spaces in the plaintext are ignored. Spaces may often be removed before processing the text for a message.

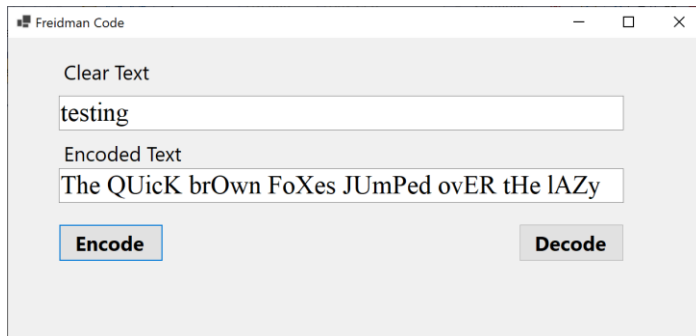
Cryptography Issues

A cipher like this works best if the differences are not too obvious to the casual observer. The big disadvantage of using mixed case letters is that it is not very subtle. It's almost obvious that there is something special going on. Italics are less noticeable. Serif and Sans Serif are even more subtle.

This is not the sort of encryption that holds up to serious attack. It relies almost completely on obscurity; of people not knowing there is a code at all.

Project Suggestions

Create a program to embed a message in a plaintext using a mix of uppercase and lowercase letters. The program should be able to decrypt the message as well. The program requires a plaintext message where the secret message will be hidden. The plaintext must have 5 times as many letters as the secret message. This is a sample program screen.



One important piece of description is determining if a letter is uppercase or lowercase. In some systems, Microsoft's .NET Framework for example, there is an `IsUpper` method that will return true for uppercase letters or false for lowercase letters. If you do not have such a method in the programming environment you are using there is some information in the sample code chapter that may help. See **Determine if a letter is uppercase or lowercase**.

Book Ciphers²⁴

Introduction

There are several variations of the book cipher. The common thread of all of them is that a common text is used as the basis. Generally, this is a book or part of a book that is in common usage. It should be source that doesn't appear to be unusual when found with either of the people involved in the secret message. The Bible is a frequent choice. What ever book is used, both individuals must have the same book in the same edition. For example, both parties might use the King James version of the Bible. Or perhaps the same printing/edition of a novel. The proper functioning of a book cipher is that a reference to a word or letter in one copy of the book is the same as the copy the other party has.

The simplest form of a book cipher is that a number or set of numbers identifies a specific word that is both in the text and in message. This requires that all of the words in the message to be encoded exist in the document being used in the process. That can be a problem for some types of messages. Also, the same word from the message should be replaced with different instances of the word from the code source document to make breaking the code more difficult. Using multiple instances of a word is a form of Homophonic substitution²⁵.

This type of book cipher is technically a code not a cipher.

A type of book cipher that gets around the limitations of the simple book code is one that replaces each individual letter with a reference to a word such that the first letter of the word is the same as the letter being replaced. This can result in long encoded messages. This is a cipher and not a code.

Encrypting

A code book is required. It must include all the words you want to encode if you want to use a simple book cipher. Next, search the source for an occurrence of the word and note its location. With a hard copy book the location may include the page number, the line number, and the location in the line. If using a text file the line and word in the line will be sufficient. Note that increased security may be attained if the search starts are a specific line or page that has been agreed to in advance.

Build a message by using the locations of the words in the code book.

If you are creating a cipher where each word is spelled out, you also need a code book. The process for encrypting the message is like the previous simple method except that the words used are dependent on the first letter of the word. For added security, save the location of the word used for each letter so that the next word search starts after the last word used.

A computer program might make a sort of dictionary that lists the words with their locations in alphabetical order. Arrays of locations for each word or letter, in the case of a true encryption, might make encryption software easier and more efficient if Homophonic substitution is a goal.

²⁴ https://en.wikipedia.org/wiki/Book_cipher

²⁵ https://en.wikipedia.org/wiki/Substitution_cipher#Homophonic_substitution

Decrypting

Decrypting a message is as simple as opening the proper book and finding the pages and words or letters identified by the numbers received. Each item identified is recorded until the entire message has been decoded.

Cryptograph Issues

Decrypting requires that both parties have the same book printed in the same format and the same edition. Spacing can vary between editions and printings. Keeping the book used a secret is essential. If the message is intercepted the first step to breaking the encryption is to identify the book used. For this reason, the book must be carefully chosen as one a third party would not easily identify as a possible key.

Project Suggestions

A good online project is to have students send messages encrypted and encrypted using actual books. A textbook or assigned reading book common to both students is an obvious possibility.

Creating a program to build encrypted messages from a known text file is a possible project. A pair of students might work together with one student creating an encrypting program and the second creating the decrypting program. This provides students with practice reading and parsing text in files. Project Gutenberg²⁶ is a great source for large files that may be appropriate for this project. They have some 75,000 public domain books available as text files.

²⁶²⁶ [Free eBooks | Project Gutenberg https://www.gutenberg.org/](https://www.gutenberg.org/)

Playfair cipher²⁷

Introduction

The Playfair cipher is a substitution cipher that encrypts pairs of letters unlike some other ciphers we have looked at which encrypted individual letters. The cipher was invented in 1854 by Charles Wheatstone, but is called the Playfair cipher after Lord Playfair who promoted its use. Because there are 600 bigrams (possible pairs of letters) rather than 26 letters in the English alphabet it is harder to break than single letter substitutions. Also, frequency analysis does not work well as it would in simple single letter substitution ciphers.

Modern computers can break these ciphers quickly, but this was a secure cipher in its time.

Encrypting

The first steps involved in the encryption process are the creation of a 5x5 Polybius Square starting with a key phrase. Normally the letters I and J are used as synonymous. Using the keyword "PLAYFAIR" the square would look like this:

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

The plain text message must be broken down into pairs (digraphs). If an odd number of letters are in the plain text an "X" or a "Z" is used to even the count. The plain text "TESTING" would be encoded as "TE ST IN GX". If letters are duplicated then add the letter "X" after the first letter. The plain text for "HELLO WORLD" would be examined as "HE LX LO WO EL DX" The actual encoded text would omit spaces.

1. If the pair has the same letter twice (or there is only one letter left by itself at the end of the plaintext) then insert the letter "X" **between** the duplicate letters (or at the end), and then move on to the next step.
2. If the two letters appear on the same row in the square, then replace each letter with the letter immediately to the right of it in the square (wrapping around to the left-hand side if necessary).
3. If the two letters appear in the same column in the square, then replace each letter with the letter immediately below it in the square (wrapping around to the top of the square if necessary).
4. Otherwise, form the rectangle for which the two plaintext letters are two opposite corners. Then replace each plaintext letter with the letter that forms the other corner of the rectangle that lies on the same row as that plaintext letter (being careful to maintain the order).

²⁷ [Playfair cipher - Wikipedia https://en.wikipedia.org/wiki/Playfair_cipher](https://en.wikipedia.org/wiki/Playfair_cipher)

Decrypting

The first step in decrypting a message is to create a 5x5 Polybius Square starting with a key phrase. The square, and the key phrase, must be identical to the key phrase used to encrypt the ciphertext.

Next, split the message into pairs of two letters (digraphs). For each pair of letters in the ciphertext, you apply the following rules in reverse to find their corresponding plaintext letters.

1. If the letters are in the same row, replace each letter with the letter immediately to its left in the square. If a letter is at the leftmost position, wrap around to the rightmost position of the same row.
2. If the letters are in the same column, replace each letter with the letter immediately above it in the square. If a letter is at the topmost position, use the letter at the bottom of that column.
3. If the letters form a rectangle (not in the same row or column), this rule remains the same as encryption. Form a rectangle with the two letters and take the letters on the opposite corners of the rectangle that lie on the same row as the ciphertext letters.
4. After decryption, you will need to manually remove any filler characters (commonly 'X' or 'Z') that were inserted during encryption to separate double letters or to make the plaintext an even length. These filler letters are usually identified as they do not fit the context of the message.

Once the message is decrypted the starting and ending of words in the message may require some effort.

Cryptography Issues

Modern computers can break the encryption using brute force methods by matching the bigrams (pairs of letters) with the known frequency of bigrams of the original language. Frequency analysis of the bigrams is very helpful in this regard. While the Playfair cipher is harder to brute force crack than some other letter substitution ciphers, if a large set of ciphertext is available the combination of high speed computers and letter frequency analysis makes the Playfair cipher breakable.

The five-by-five grid used in this cipher also limits what characters can be used. For example, no numbers or special characters can be used. Language that depends on both upper- and lower-case letters are also problematic.

Project Suggestions

One possible starter project would be to create a 5x5 Polybius Square starting with a key phrase.

A more involved project would be to create a program to encrypt and decrypt using the Playfair cipher. In a group project, one student might be responsible for generating the square, another student to create an encryption routine, and a third student to create the decryption routine. The second and third students would use the create square routine created by the first student. A fourth student might be responsible for creating the main routine incorporating the work of the other students. This idea has the advantage of helping students understand the need to work together on interfaces to routines.

Playfair Cipher

Plain Text

Do not tell anyone about this

Key Word

playfair

Encrypted Text

RTOQSZNMVYVAYSPQOHPRQZNQMCN

Encode

Decode

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

Display Matrix

Useful Links

- [Playfair Cipher with Examples](https://www.geeksforgeeks.org/playfair-cipher-with-examples/) <https://www.geeksforgeeks.org/playfair-cipher-with-examples/> Includes code implementations to encode and decode the Playfair Cipher in C
- [Playfair Cipher \(online tool\)](https://www.boxentriq.com/code-breaking/playfair-cipher) <https://www.boxentriq.com/code-breaking/playfair-cipher>
- [Playfair Cipher - Crypto Corner](https://www.interactive-maths.com/playfair-cipher) ([interactive-maths.com](https://www.interactive-maths.com))

Code Samples

Introduction

These samples were written and tested using the C# programming language and are provided for reference with no guarantee they will work in your program or programming language.

Padding a string with random letters

```
private string padString(string input, int size)
{
    /* Create a string padded the a requested length
    * Inputs: Original string & Length of desired output string
    * Assumes adding to a string
    * Returns: New string with random letters as padding to length requested
    */
    string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    Random R = new Random();

    for (int i = input.Length - 1; i < size - 1; i++)
    {
        input += alphabet.Substring(R.Next(alphabet.Length), 1);
    }
    return input;
}
```

Creating an array of numbers in a random order

```
private int[] BuildKeys(int length, int seed)
{
    /*
    * Builds a key string
    * Inputs: Length of key to create & Seed for random number generator
    * Output: integer array to be used as a key for encoding and decoding
    */
    Random r = new Random(seed); // Use seed value to start generator
    int[] working = new int[length]; // Create an array of the required length
    for (int index = 0; index < length; index++) // Initialize key array
    {
        working[index] = index;
    }
    // Shuffle the key array
    for (int index = 0; index < length; index++)
    {
        int temp = working[index];
        int swap = r.Next(length);
        working[index] = working[swap];
        working[swap] = temp;
    }
    return working;
}
```

Determining the ordinal position of a letter

For English letters, the position in the alphabet of a letter can be done easily by taking advantage of the ASCII codes. Using a Char data type the integer value of the char subtracted by the integer value of the letter 'a' or 'A' results in what position in the alphabet. This example assumes that c is a variable of type Char.

```
int x = (int)c - (int)'a';
```

An alternate way is to create a string of the letters in order and search for the position of your letter in that string. Assuming s is a string variable with a single letter stored in it.

```
string alphabet = "ABCDEFGHIIJKLMNOPQRSTUVWXYZ";  
int location = alphabet.IndexOf(s);
```

Building a String Representation of a Binary ASCII value

```
string OnesAndZeros(char c)  
{  
    /* builds as string of ones and zeros that represent the Binary ASCII value of a char  
    * The Binary AND operator (&) is used to check the bit settings within the char  
    */  
    string output = "";  
    if (((int)c & 128) != 0) output += "1"; else output += "0";  
    if (((int)c & 64) != 0) output += "1"; else output += "0"; ;  
    if (((int)c & 32) != 0) output += "1"; else output += "0"; ;  
    if (((int)c & 16) != 0) output += "1"; else output += "0"; ;  
    if (((int)c & 8) != 0) output += "1"; else output += "0"; ;  
    if (((int)c & 4) != 0) output += "1"; else output += "0"; ;  
    if (((int)c & 2) != 0) output += "1"; else output += "0"; ;  
    if (((int)c & 1) != 0) output += "1"; else output += "0"; ;  
    return output;  
}
```

Determine if a letter is uppercase or lowercase

If you do not have such a method available there is an easy way to determine if an ASCII value is uppercase or lowercase. In ASCII, there are 32 characters between an uppercase letter and a lowercase letter. For example, the uppercase A is 65 and the lowercase a is 97. This means that the 32 bit is set, or 1, for lowercase letters and unset, or 0, for uppercase letters. Many languages support operators that support bit operations. In many languages the & sign [single & not to be confused with && which is a Boolean AND] that lets a programmer compare values at the binary level. The following code uses the & operator to see if the 32 bit is set.

```
private Boolean isUpper(char c)  
{  
    Boolean retCode = false;  
    int check = c & 32;  
    if (check == 0) retCode = true;  
    return retCode;  
}
```

Appendix

Resource Links

You may find some of these sites helpful:

- [Cryptography Hints \(nd.edu\)](#)
- CS Unplugged: <https://classic.csunplugged.org/public-key-encryption/>
- [Caesar Cipher Lesson \(ncyte.net\)](#)
- [OpenComputerScience - 4. Information & Cryptography \(opencompsci.com\)](#)
- [C5 Lessons \(ncyte.net\)](#)

Frequency of Letters in English²⁸

e	13.0%	m	2.4%
t	9.1%	w	2.4%
a	8.2%	f	2.2%
o	7.5%	g	2.0%
i	7.0%	y	2.0%
n	6.7%	p	1.9%
s	6.3%	b	1.5%
h	6.1%	v	1.0%
r	6.0%	k	0.8%
d	4.3%	j	0.2%
l	4.0%	x	0.2%
c	2.8%	q	0.1%
u	2.8%	z	0.1%

²⁸ https://en.wikipedia.org/wiki/Letter_frequency#Relative_frequencies_of_letters_in_the_English_language

Non-English Alphabets

https://en.wikipedia.org/wiki/Letter_frequency#Relative_frequencies_of_letters_in_other_languages

Greek Alphabet²⁹

A α, B β, Γ γ, Δ δ, E ε, Z ζ, H η, Θ θ, I ι, K κ, Λ λ, M μ, N ν, Ξ ξ, O ο, Π π, P ρ, Σ σ/ς, Τ τ, Υ υ, Φ φ, Χ χ, Ψ ψ, and Ω ω.

Norwegian Alphabet³⁰

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	Æ	Ø	Å
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Turkish Alphabet³¹

A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Hawaiian Alphabet³²

a	e	i	o	u	h	k	l	m	n	p	w
---	---	---	---	---	---	---	---	---	---	---	---

Yes, only 12 letters!

²⁹ https://en.wikipedia.org/wiki/Greek_alphabet#Letter_shapes

³⁰ https://en.wikipedia.org/wiki/Danish_and_Norwegian_alphabet

³¹ https://en.wikipedia.org/wiki/Turkish_alphabet#Letters

³² <https://www.interestingfacts.org/fact/hawaiian-alphabet>