# Monoalphabetic Cipher Research

## **Introduction / Background**

During my research I wish to gain an insight into some tactics for decrypting a monoalphabetic cipher. I am sure that there is already an implemented software solution however this project is for my own personal learning, and I am aiming to try developing a solution without researching any explicit implemented code.

Note that this Is not an academic piece, while references are included, they have not been implemented in a formal academic matter.

## **Initial thoughts**

In total, there are 26! Combinations of possible combinations to decrypt, my aim is to reduce this amount as much as possible, then sort the last remaining possible combinations by most likely, manually confirming each. 26! Is a massive number, ideally, I would be looking to manually sort less than 100 combinations which equates to around 5 unknown letters? However if a system was created where letters within a combination can be accepted, the number of combinations to be sorted would be significantly less as I would imagine that the process would be quick to confirm.

This sort of tactic can be seen in several board games such as mastermind, I believe it would be beneficial to analyse the tactics used when playing that game. The development of the morse code system would also be beneficial to look out, as I imagine that common letters were kept simple to increase communication efficiency.

On the initial surface, there are several tactics that I can think of that can initially reduce our search space, these are briefly described as follows.

* Eliminating uncommon letters (x,z,..)
* Placing common letters (vowels)
* Using common words (the, and, conjunctions)
* Using common patterns within the English language (I before e, s occurring at the end of plurals)

## **Research**

Letter Frequency

The labelling of letter frequency depends on multiple factors, such as the topic at hand, the type of English used and academic vocabulary of the sender. Within morse code, messages are kept to the point to ensure efficiency whilst an extract from a novel contains an array of words that may not be commonly used. The debate between the US and British English may also come into play, with the common swap of the letters z and s, affecting which letter is the most common. Across multiple sources, there has been multiple studies conducted around the frequency of letters within a text extract. These are generally placed into two categories, dictionary and general English. Percentages do not differ between sources by much, so here is a table presented in relation to cipher decryption. (Funnily enough, I found this table before the cipher encryption-based website!)

<http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html> (table)

<https://www.grammarly.com/blog/rarest-letter-in-english/> (background knowledge)

This could be implemented in a simplistic matter, guessing the most common letters, and ignoring the least common letters.

Common words

Common words can be used within the decryption, especially in relation to a specific topic and common short words. If the individual trying to decrypt knows the topic of the message, they could enter specific words that they expect to come up within the message, once initial letters have been found through letter frequency, these words can be specifically searched for. Smaller common words can be used to find and confirm more letters within the message, these can be conjunctions or general smaller words.

**Single letter words:** a, i

**Two letter words:** of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am

**Three letter words:** the, and, for, are, but, not, you, all, any, can, had, her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way, who, boy, did, its, let, put, say, she, too, use

**Four letter words:** that, with, have, this, will, your, from, they, know, want, been, good, much, some, time

These common words were found mostly in relation to cryptography which is pretty cool! These words were taken from the same website as before just from a general google search, here Is the link.

<https://www3.nd.edu/~busiforc/handouts/cryptography/cryptography%20hints.html>

Common Patterns:

There are also a number of common patterns within the English language that we can use to our advantage, this includes double letters, diagraphs and trigraphs.

**Double letters:** ss ee tt ff ll mm oo

**Digraphs**: th er on an re he in ed nd ha at en es of or nt ea ti to it st io le is ou ar as de rt ve

**Trigraphs:** the and tha ent ion tio for nde has nce edt tis oft sth men

Whilst double letters can be easy to implement, it can be hard to implement some of the diagraphs shown, I would recommend implementing this technique last if the others do not provide a solution that meets the requirements.

## **Confirming the decryption:**

To create a system that is efficient, we must present an efficient interface that the user can use to confirm if the decryption is correct. One method that I can visualise is the program presenting single words at a time, where the user can confirm or deny whether it is correct, otherwise stating letters that the word may include. -edro, well that has the letter P

## **Thoughts on implementation**

After some on paper practice, it seems to be harder than I thought to perform the decryption in a logical model without implementing some heavy decision-making processes along the way. You might be able to implement a searching algorithm that sorts through different combinations, however for the moment this is out of my knowledge area (future updates…). However, I have come up with a method that can be guided with the help of the user throughout the process, this can be seen as an assisting decryption software.

My basic implementation is turned based, using the users input to confirm or deny the software attempts. This will run through the following processes for each turn, restarting once a character has been suggested.

* Letter Frequency, looking for distinct large frequencies. (This will commonly find the most frequent characters)
* Checking words that have been decrypted more than 40% for keyword matches.
* Checking words that have been decrypted more than 60% for user guesses.
* Single letter words, this will terminate once they have all been decrypted.
* double letter words, this will terminate once they have all been decrypted.
* triple letter words, this will terminate once they have all been decrypted.
* quadruple letter words, this will terminate once they have all been decrypted.
* Common sounds.
* Finally, indicating to the user that the software is stuck.

## **Other Notes:**

Accidently stumbled across a good link, not looking I promise.

<http://www.richkni.co.uk/php/crypta/index.php> (text decryption software)

<http://pi.math.cornell.edu/~mec/2003-2004/cryptography/cryptography.html> (cryptography background)