Coursework Report

Steven Pyper

40319882@napier.ac.uk

Edinburgh Napier University – Web Tech (SET08101)

Contents

[1 Introduction 2](#_Toc2855433)

[1.2 Caesar Cipher and Rot13 2](#_Toc2855434)

[1.3 Direct Substitution Cipher 2](#_Toc2855435)

[1.4 Morse Code 2](#_Toc2855436)

[2 Software design 2](#_Toc2855437)

[2.1 Pages 2](#_Toc2855438)

[2.1.1 Home Page(index) 2](#_Toc2855439)

[2.1.2 Caesar and Rot 13 2](#_Toc2855440)

[2.1.3 Direct Substitution Cipher 3](#_Toc2855441)

[2.1.4 Morse Code 3](#_Toc2855442)

[2.2 3](#_Toc2855443)

[References 4](#_Toc2855444)

[Appendix 5](#_Toc2855445)

# 1 Introduction

The main scope of this coursework will be to build a website that contains at least 2 ciphers. These will take inputs from the user that allow them to specify the key(if the cipher takes one) and the text they would want to convert. These must also be reversable so that the user can uncypher the text after they have ciphered it.

## 1.2 Caesar Cipher and Rot13

The ciphers I have chosen to do are a simple Caesar cipher. From reading (wikipedia, n.d.) about this cipher it should be somewhat simple to implement but also is a useful cipher that can be explained well to people who aren’t tech savvy, this allows it to be a good first option for the website as a introduction to ciphers as well as a good introduction to JavaScript due to it in theory not being complicated. In addition to the Caesar Cipher the user will be allowed to use a rot13 cipher which is essentially just the Caesar but with a fixed shift of 13, this will be made after the Caesar as it should take less code to go that way rather than reverse it, this is another simple cipher for the user to get introduced to and is there for a good early choice.

## 1.3 Direct Substitution Cipher

Another cipher I will implement is a direct substitution cipher, I believe that this will be a more complicated cipher but also will be much more interesting in terms of what the user would be able to do with it. As it relies on the user specifying their own alphabet(key) this will allow them to choose their own alphabet to start with, allowing it to work with more than just English(like the Caesar Cipher) and then create their own, but also allow them to cipher it multiple times and as long as they have all the keys in the right order get back to where they started after multiple deciphers (Wikipedia, n.d.). Due to all of this it should be a more difficult cipher both to code and for the user to understand, while researching about this and realising that it will be difficult for the user to understand I have decided that although all ciphers will be available on the first page, each cipher will have its own page explaining how to use the cipher and how it works. This will allow the user to easily understand how to use the cipher and maybe even learn more about it.

## 1.4 Morse Code

I will also implement Morse code this is the wildcard out of all the other ones as its technically not a cipher but is still very interesting and fits with the idea of the website of converting text, this also allows for the use of audio as it should be possible to play the sound of the Morse to the user through the browser. This should be the most difficult to implement as it will have to deal with time between each character and the time between each word. This should still allow the user to convert between text and Morse and play audio going both ways(so if the user enters Morse to convert to text, or enters text to convert to Morse). I would like to try to stick to standard (Wikipadia, n.d.)(See Appendix A) however if this would take to much development time as long as it is very clear what the difference between dots, dashes, spaces between characters, and spaces between words I will be happy with the implementation.

# 2 Software design

The initial design will include just one page(index.html) which will contain all the ciphers and information about them. This information will contain a little idea behind the cipher and what it does, with the possibility of having a little explanation of how to use it. This will also include an image that might help the user to understand what the cipher does by showing what it does as well as writing what it does. This however most likely will be placeholder and only be used as a testing page which will allow me to see how much room will be needed as well as make changes to how the output will be displayed.

There will be a page for each of the ciphers except for the Caesar and rot13 which will be together as they are very similar and can be explained together which will help the user with the simple version first as well as making it tidier

## 2.1 Pages

All pages will be accessible from every other page( can go from index to anywhere and anywhere to index)

(See Appenix B for NavMap)

(See Appendix C for home page layout)

(see Appendix D for Other Page Layouts)

Unsure where the Navigation bar will go, it will either go straight down the left side of the page or below the heading depending on which looks nicer at the time of creation, therefor none of the layouts include the navigation bar!

### 2.1.1 Home Page(index)

The final design will be to have the first page only explain some of the strengths and weakness and the history of each cipher, this page will then contain links to each of the individual ciphers which the user would be able to choose either from a navigation bar or at the end of the explanations of the ciphers. This will allow the user to then decide what cipher they might want to use. This page will most likely contain 3 rows and columns, one row for each of the ciphers (with Caesar and rot13 together) that way everything can be split up in a way that makes sense, the name will include a link to the page that has the cipher. One column will be used as the heading for the cipher name and will also give little run down about the cipher which will include what it does and a little history about it. The next column will include the pros of using that cipher as well as good use case scenarios for using it. The final column will include the negatives of the cipher and the ways that it can be easily broken (both in terms of user input and by cracking methods).

### 2.1.2 Caesar and Rot 13

Due to how similar these two are they will both be put on one single page. Rot13 can help explain the Caesar cipher and as are likely to either use the same base coding, or similar, so it makes sense to keep them together on one page. The page will consist of two rows and two (with an extra row which will be single column for the output). The first row and column will include information about the cipher and how it works, this will include information like how it moves the alphabet by a certain amount. The second column will include how to use the cipher which will be dependent on how it ends up being coded. For instance, the key may need to be converted to negative by the user when they want to decode, or I could make the code deal with this while decoding. This will help the user with how to use the cipher and explain what each input box and button will do. On the second row the first column will include a selection box (drop down menu) which will allow the user to decide between whether they would like to encrypt or decrypt their message. Depending on what the user selects different messages will appear in the next column which will ask the user to enter information and then click a button which will then run the inputs through the program. The outputs from the program will be displayed on the extra row.

### 2.1.3 Direct Substitution Cipher

This page will follow the same style as the Caesar and rot13 page in terms of layout however the information will include more detail about helping the user with the cipher as it is more complicated due to needing more inputs than the other ciphers. It will allow the user to encrypt their text based off the original alphabet they enter (which will most likely be the full English alphabet) and the new version of the alphabet (their own modified version), this will be the same for decrypting where they will enter the same information but getting the decrypted text back.

### 2.1.4 Morse Code

This page will also follow the same style as the previous ciphers however will include extra buttons while converting which will have a button to play audio (as well to stop it from being played). This page will include information about Morse code and tell the user if it applies to standards or not, as well as how to use it. There will be a choice between converting to Morse and converting to text, and during both there will be an option to play either the text (which will be converted to Morse) or the Morse that was entered as audio output as well as text output.

## 2.2 JavaScript

there will be a script for each cipher on the website and there will also be a script that will allow the user to decide how they would like to use the cipher (encrypt or decrypt). On the test index there will also be a script that will allow the user to decide between each cipher however this will most likely go unused in the final version as each script would be on its own page meaning that the user won’t need this functionality.

### 2.2.1 Caesar Cipher and ROT13

This should be a fairly simple program to write due to the fact that I have wrote similar programs before, from this I know that there are two ways to do it, one which is referenced on the Wikipedia page for Caesar ciphers is to do it numerically, this method means using maths to calculate what the remainder would be from adding the original character number(with a being 0) with the key, this would mean that if the character z was shifted by 1 it would become the number 26 which would leave no remainder so the number would become 0 which would reference a. this is a system that does work however the flaw is that it can end up being more complicated to debug when things don’t work. therefor I will be using the other method where you have one large string that is your character set (which is just all the characters A-Z), using this it is possible to get a character form the user, get where it would be in the character set(so a would still be 0) and you can then apply the key to it, this allows you to have more control over what the script should be doing depending on what number the character would be converted to, as an example the number could go below 0 or above 25 and these would need to be dealt with differently. This script will have 3/4 different functions, this will be dependent on if there will be a “main” function which will choose which type you are using or if there will just be the functions for dealing with encryption, decryption and cracking (where the key is not known). The reason for keeping the main function would be that you could have the button for rot13 still call up the Caesar function but pass in a key of 13 instead of asking the user to enter a key.

### 2.2.2 Substitution Cipher

The substitution cipher should be more difficult than the Caesar as you are having to deal with two different alphabets, one which will be the original alphabet (the English alphabet for example) and the modified alphabet which user can define themselves. It will include functions for encrypting and decrypting the messages although they will likely be very similar but with different references. The simplest way I found to two link two alphabets together would be the use of maps (alligator.io, n.d.). this should allow me to have a function that will link the two separate alphabets together and allow for the program to look through each character entered, compare it to the original alphabet and then find the value that would be returned from that key in the map. This should allow for an efficient and fast way to take the users inputs and convert them to the new alphabet. There may need to be more functions to change the alphabet depending on if its being encrypted or decrypted although I aim to only have one function doing all the setting up of the map

### 2.2.3 Morse Code

The Morse code JavaScript will most likely be the most complicated one of them all, this is because it is not a standard switch where you convert one character into another single character. It will mean that you must convert a single character into multiple, for example “a” would become “.-“ , this becomes further complicated when you add multiple characters and then multiple words as there has to be spaces between characters but also spaces between words so you will have to distinguish between the two, this will impact the code heavily as it will need to know what the difference is between them so that when it converts it knows where each letter starts and ends, where each word starts and ends and where actual spaces should be. There is also the added difficultly when then converting this to audio, there is a standard in place of how long each sound should last which must be represented in the code, this means learning about the audio api and just getting a sound to play and then getting it to play and stop when it should, the aim will be to have it follow the standard but if that is not possible then I will get it to a point where it is fairly easy for the user to figure out the difference between characters, words and true spaces between words. Getting the text converted to Morse and then back to text will greatly benefit from using maps like the substitution cipher, this is due to the fact that we have two predefined languages to convert between, it will be possible to have a map that links the exact Morse linked with its respective English character (“.-“,”A”) and the opposite is also true, therefor two maps will likely be created that are opposites of each other, one will be used when converting to text and the other when converting to Morse. This should help simplify some of the converting as it means the main chunk of code will be used to parse what spaces should be and where each character starts and ends. Finally, for Morse there will have to be a function dedicated to playing the sound, this will likely call one of the converting functions depending on what the user has input, in theory it should then be possible to take what is converted and have a loop that goes through each character and plays for different lengths depending on which character is read, from research it should be possible to do this with the setInterval or setTimeout functions. It is most likely that I will use an oscillator as it can be set to a certain frequency and by using a gain node it should also be possible to make the sound a comfortable level.

### 2.2.4 Change Cipher/ Change Method

These scripts will be used to change the information on the page depending on what the user is wanting; the change cipher script would allow the user to choose between all the ciphers (and Morse code) and the change method will allow the user to choose how they would like to use that cipher(encrypt/decrypt), the change cipher method will likely only be used for testing while there is only one page as each cipher will have its own page in the final design meaning that the user has already made that choice by going to the page. The change cipher function will add in a new selection box onto the page which contain the options for change method for the user to select (therefor if they choose Caesar there will be 3 options in the method selection box). The change method function will allow the user to choose their method and then it will add all the required information onto the screen such as new input boxes and buttons that are required for the specific task. This will be required on each page that has a cipher.

# 3 Implementation

## 3.1 Implementation of design

PLACEHOLDER

## 3.2 implementation of Code

### 3.2.1 Implementation of Caesar Cipher and Rot 13

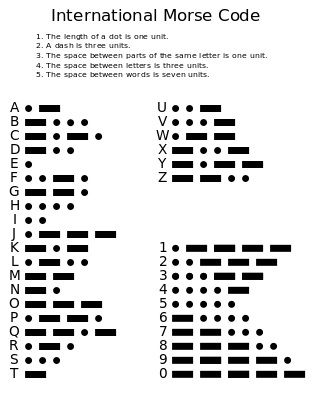
# References

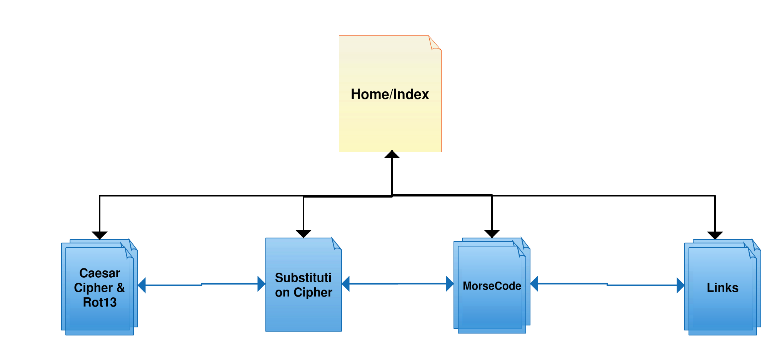
Wikipadia. (n.d.). *https://en.wikipedia.org/wiki/Morse\_code*. Retrieved from https://en.wikipedia.org.

wikipedia. (n.d.). *https://en.wikipedia.org/wiki/Caesar\_cipher*. Retrieved from https://en.wikipedia.org.

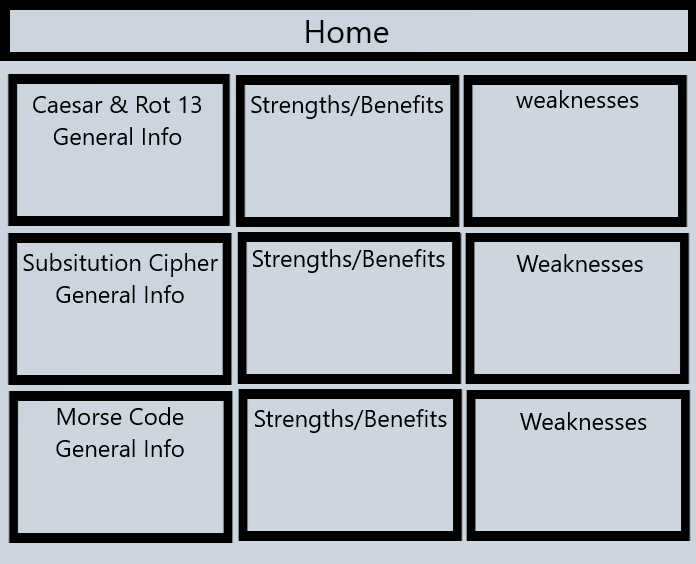
Wikipedia. (n.d.). *https://en.wikipedia.org/wiki/Substitution\_cipher*. Retrieved from https://en.wikipedia.org.

# Appendix



Appendix A - (Wikipadia, n.d.)

Appendix B – (created on Creately.com)

Appendix C –

Appendix D - 