

Caesar Cipher and Vigenère Cipher
Christina Hartnett
110331535
AMS 595 Final Report
December 2020

1 Introduction and Objectives

1.1 Background

Cryptography and cryptanalysis are techniques that have been used since ancient Greek times. They were created to protect secret messages or information. Prior to the 20th-century, encryption and decryption methods were done with pen and paper. Oftentimes, these techniques were used during wars. In World War I, the United States was able to decrypt a German telegram prompting them to join the war. In World War II, the United States was able to shorten the war with the decryption of Nazi ciphers. Many more advanced techniques have come into existence since the first ancient ciphers. This report and code contains two different types of ciphers; the Caesar shift and the Vigenère cipher.

The Caesar cipher is one of the oldest and simplest methods of encryption. This method is a shift substitution cipher. This method was famously named after Julius Caesar who often used this simple method to send private correspondence. This method is not often used today as it is easily deciphered by a brute force attack. It is simple to break because there are only 50 possible shifts (at least in the English alphabet) the words can move. Each letter in the plaintext word will be shifted by the same number of positions.

The Vigenère cipher is many small shift ciphers in one. This cipher takes two string inputs; a plaintext word and a keyword. The positions of the plaintext word will be matched to the corresponding position in the keyword. These letters are then found on the Vigenère table and the output is the new encrypted word. Each letter of the word will have its own shift based

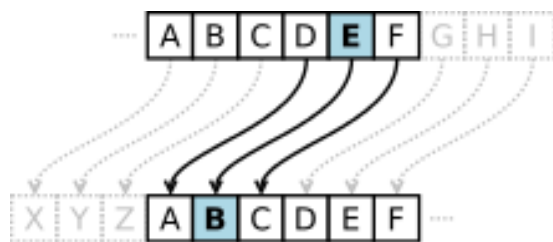
on the keyword. A Vigenère encrypted word will be a much harder encryption for a layman to decipher. Without the keyword given, the decoder would have to go through every word in the English language (approximately 173,000 words), making this decryption nearly impossible.

1.2 Goals

The goal of this report and code is to create working ciphers that will encrypt and decrypt word in two different method. The code will have a graphical user interface that will interact with the user to ask them for a plaintext word and a keyword to create the cipher.

2 Techniques and Tools

2.1 Caesar main function



For the Caesar shift cipher, I decided the technique that made the most sense was to create a dictionary. I gave each letter of the alphabet a corresponding integer, the letter was the key and the integer was the associated value. The dictionary is as follows:

```
dictionaryinitial = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8, 'i': 9, 'j': 10, 'k': 11, 'l': 12, 'm': 13, 'n': 14, 'o': 15, 'p': 16, 'q': 17, 'r': 18, 's': 19, 't': 20, 'u': 21, 'v': 22, 'w': 23, 'x': 24, 'y': 25, 'z': 26}
```

The code splits the user's string input into a list and finds the number associated with each of the letters using the aforementioned dictionary. For example, if the input is "Hello" the code will split the plaintext word into ["h", "e", "l", "l", "o"]. This list of keys is then matched to the corresponding values in the dictionary and changed to [8, 5, 12, 12, 15]. To encode the word,

the user must input a key which is an integer between -25 and 25. If the input is negative, the shift is to the left and if the input is positive the shift is to the right.

In practice, I took this integer and added it to each of my dictionary values. When I did this, I realized that the dictionary values would go above 26, which was not the intended result. To correct this, I created an “If” statement that said associated if the value in a dictionary was greater than 26, we would adjust it to make it a value between 1 and 26. As an example, if my shift made the letter “z” correspond to a 27 the new adjusted shift would have “z” correspond to 1. Once this was complete, I matched the list of numbers derived earlier to my new shifted dictionary. I struggled with this because I could not figure out how to access the of the keys of the original list to match it back to its adjusted key. After some research, I found that the best way to go about this was to swap the keys and the values of the dictionary. To elaborate this means the key would now be an integer and the value would be a letter. I was then able to match the keys and return the value which was a new list of letters. I joined these letters together to return a string.

Decoding a Caesar shift cipher it is very similar to encoding, just in reverse. For this section, the user would input an encoded-word and a hint. This hint, or the shift, will tell the code that two characters are equal, for example A=D. To figure out what the shift is I changed the hint input characters to the values associated with the dictionary and then subtracted key2 from key1. This subtracted value is now the shift I will perform to the word. I created a new dictionary that adjusts the values by subtracting the number shifted to it. Next, I split the user input word into a list and matched each letter to my new dictionary. I then matched these numbers to my original dictionary, in the same manner I used in the encryption code to find the decrypted word. This was a simple code to write after the encryption code was implemented as the code just works in

reverse when decrypting. This further shows how simple it is to crack Caesar Shift ciphers because there is not a wide variety of options for the shift. Due to the relatively finite number of options, an attacker could use brute force to solve this cipher and try every shift of the 25 letters to find which makes the most sense fairly quickly.

2.2 Vigenère encryption main function

For the Vigenère main function, I struggled to find the best way to create a Vigenère table. A Vigenère table should look something like this:

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

The Vigenère table is a key component to finding the new encrypted letter. I decided to create a pandas data frame using `string.ascii_lowercase` to fill the data frame with the alphabet. I also labeled the rows and columns with the letters of the alphabet to make the table easily accessible. In order for this method to work, the keyword has to be at least the same length as the plaintext word. I created an “If” statement that checked the length of the word and would then change the length of the keyword if needed. Both the plaintext and keyword were converted into lists. Each value in these lists became the row and column labels I used to find the new encrypted word. Then, I used the `loc` method within pandas to find the corresponding element to each row and

column label and returned the new list. Lastly, I changed this list into a string and then returned it for the user to see. As previously stated, the Vigenère Cipher is much more complex than a Caesar shift Cipher. This is exemplified by my inability to successfully implement the decryption of a Vigenère cipher in a code. A Vigenère cipher is very difficult to crack unless the keyword is known, which if a secret message is intercepted in a war the group trying to crack the code is unlikely to know the set keyword. There are hundreds of thousands of words and letter combinations in the English language whereas in the Caesar shift there are only 25 ways the word can be shifted.

2.3 Tkinter Function

For the user interface, I created a Graphical User Interface (GUI) using the Tkinter package in Python. I created an interface for the user to type in their inputs and easily be returned an output by clicking the button. I was not aware of the Tkinter package prior to this project and found how useful it can be in making a project more appealing to the eye and user friendly. I created a title that included the name and how to use it. I also created different input lines that would ask for the word and key which would then apply the correct code to them to return the answer. I also created three buttons; Result, Reset, and Exit. The result button will return either an encrypted or decrypted word. The reset button will reset the input lines. The exit button will exit the program.

2.4 Testing

To test my codes, I used many different trials and errors. I was able to use online Caesar Shifts and Vigenère ciphers to make sure my code printed the same result when the same starting values were inputted. This was a simple way to check my work. One area which required a lot of

back-and-forth testing was ensuring that my GUI was easy for a user to utilize and read. This took some effort, but after testing it many times, I believe I came to a nice format.

3 Conclusion and How to use

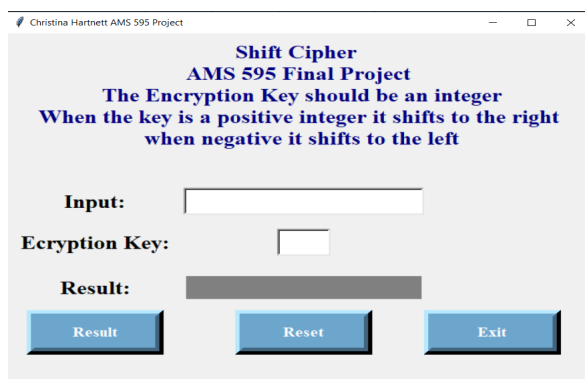
3.1 How to use: Caesar Shift Encoder

When the user runs the code, a screen will pop up with the cipher.

The user will read the instructions that state:

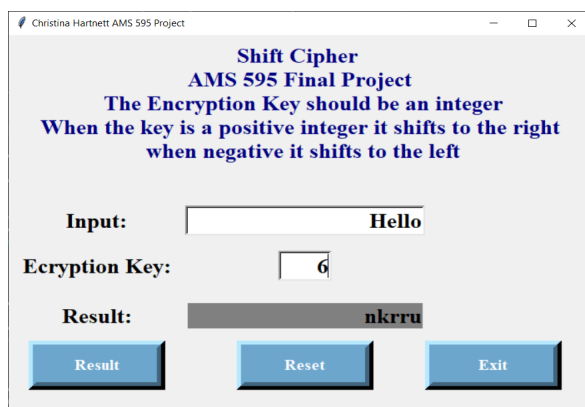
The Encryption hint should be an integer: When the key is a positive integer, it shifts the word to the right; when negative integer, it shifts to the left.

The pop-up screen is displayed below:



In the "Input" line the user should type a plaintext word. In the "Encryption key" line the user should input the integer amount they want the letters to be shifted.

Below is an example of how to use the code:

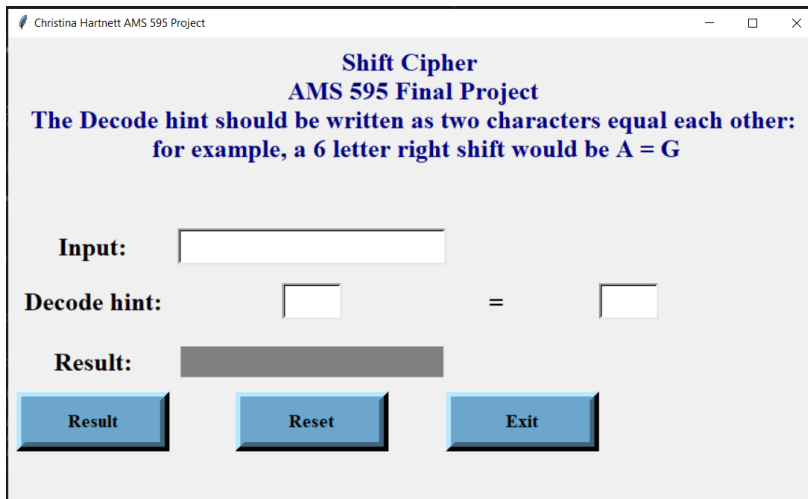


3.2 How to use: Caesar Shift Decoder

When the user runs the code, a screen will pop up with the cipher. The user will read the instructions that state:

The Decode hint should be written as two characters equal to one another: for example, a 6 letter right shift would be A = G

The pop-up screen looks like this:



Christina Hartnett AMS 595 Project

Shift Cipher
AMS 595 Final Project
The Decode hint should be written as two characters equal each other:
for example, a 6 letter right shift would be A = G

Input:

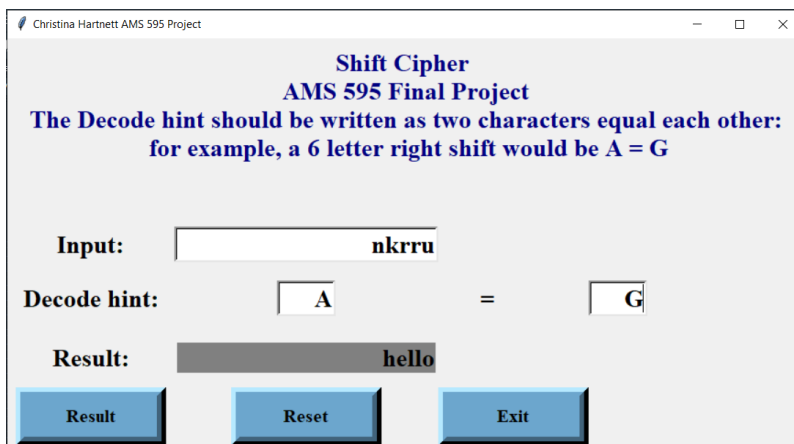
Decode hint: =

Result:

Result **Reset** **Exit**

In the “Input” line the user should type the word to be decoded. In each of the blanks for the “Decode Hint” the user should type in a letter and the difference between these letters will be the shift applied.

Below is an example of how to use the code:



Christina Hartnett AMS 595 Project

Shift Cipher
AMS 595 Final Project
The Decode hint should be written as two characters equal each other:
for example, a 6 letter right shift would be A = G

Input:

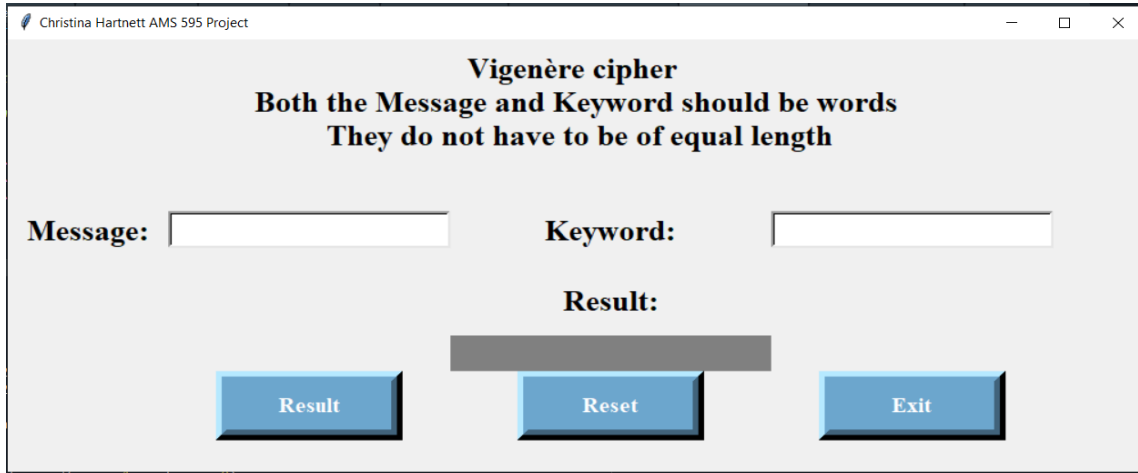
Decode hint: =

Result:

Result **Reset** **Exit**

3.3 How to use: Vigenère Shift

When the code is run a pop-up window will appear with the Vigenère cipher. The view of the user is seen below:

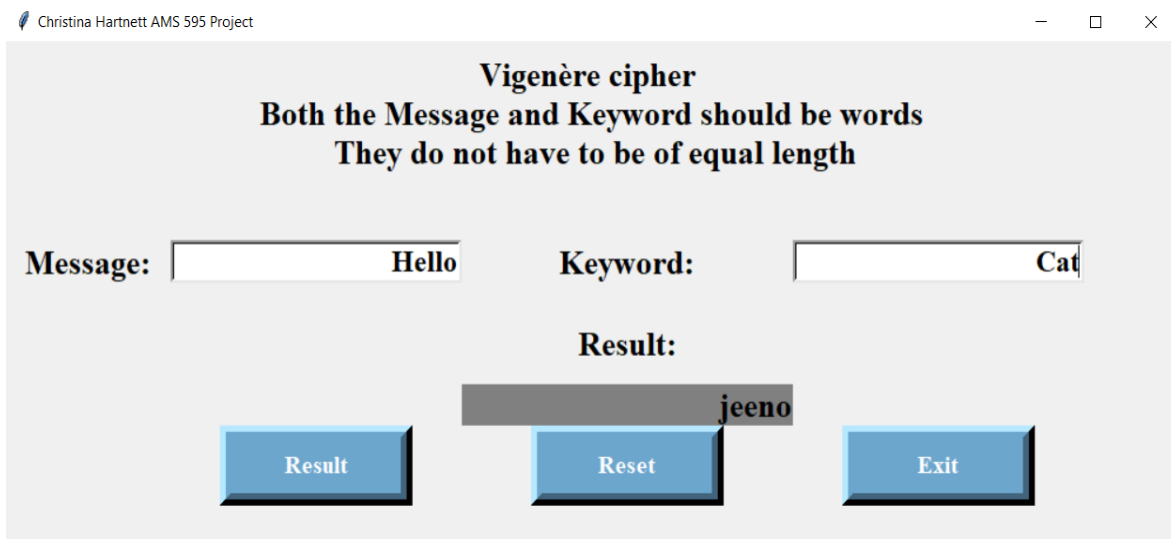


The screenshot shows a window titled "Christina Hartnett AMS 595 Project". Inside, the text "Vigenère cipher" is centered, followed by the instructions "Both the Message and Keyword should be words" and "They do not have to be of equal length". Below this, there are two input fields: "Message:" and "Keyword:". The "Result:" label is centered below the input fields, and a grey rectangular box is positioned directly underneath it. At the bottom, there are three blue buttons labeled "Result", "Reset", and "Exit".

In the "Message" line the user will input a plaintext word that they want to be encrypted.

In the "Keyword" line the user will input a word that they want the shift to be based off of.

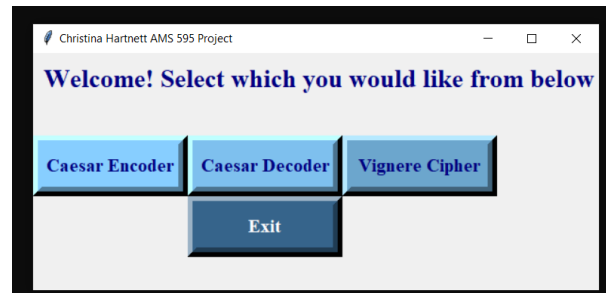
Below is an example of how to use the code:



This screenshot shows the same application window as the previous one, but with the "Message:" field containing the word "Hello" and the "Keyword:" field containing the word "Cat". The "Result:" label is still centered, and the grey rectangular box now displays the encrypted word "jeeno". The "Result", "Reset", and "Exit" buttons remain at the bottom.

3.4 Conclusion and shortfalls

I was able to implement the encryption code of both ciphers but only the decryption code of the Caesar shift cipher. The issue that I faced with the decryption in the Vigenère cipher was that the only way to simply implement the code was to make the user input the keyword they used. When intercepting a code, it is unlikely that the code breaker would know the keyword, making decryption virtually impossible. While researching, I found that many coders create a list of the most commonly used English words, but this seemed like an inefficient method. When thinking through the possibility of having the user input the key to decrypt the message, the issue I faced was trying to access a value and column in a pandas data frame and have it return the corresponding row. I was unable to find an efficient manner to perform this so I could not use this method. Another feature of the project I was hoping to implement was a main menu GUI that had the three encryption options. I was able to create this but unable to get my codes to work within it. When trying to debug I realized that when a Tkinter was placed in another Tkinter the user inputs were not recognized. I could not find a method around this and therefore decided it was best to keep all the encryptions as separate python files. I was successfully able to implement the codes I wanted and also learned a lot about the Tkinter package, dictionaries and the pandas package along the way. I am glad to have broadened my knowledge and now I can write as many secret letters as I would like.



References:

“Caesar cipher.” https://en.wikipedia.org/wiki/Caesar_cipher.

“Python GUI – tkinter.” <https://www.geeksforgeeks.org/python-gui-tkinter/?ref=lbp>.

“The Vigenère Cipher Encryption and Decryption.” <https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Base.html>.

“Tkinter — Python interface to Tcl/Tk.” <https://docs.python.org/3/library/tkinter.html>.

“Vigenère cipher.” Wikipedia, https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher.