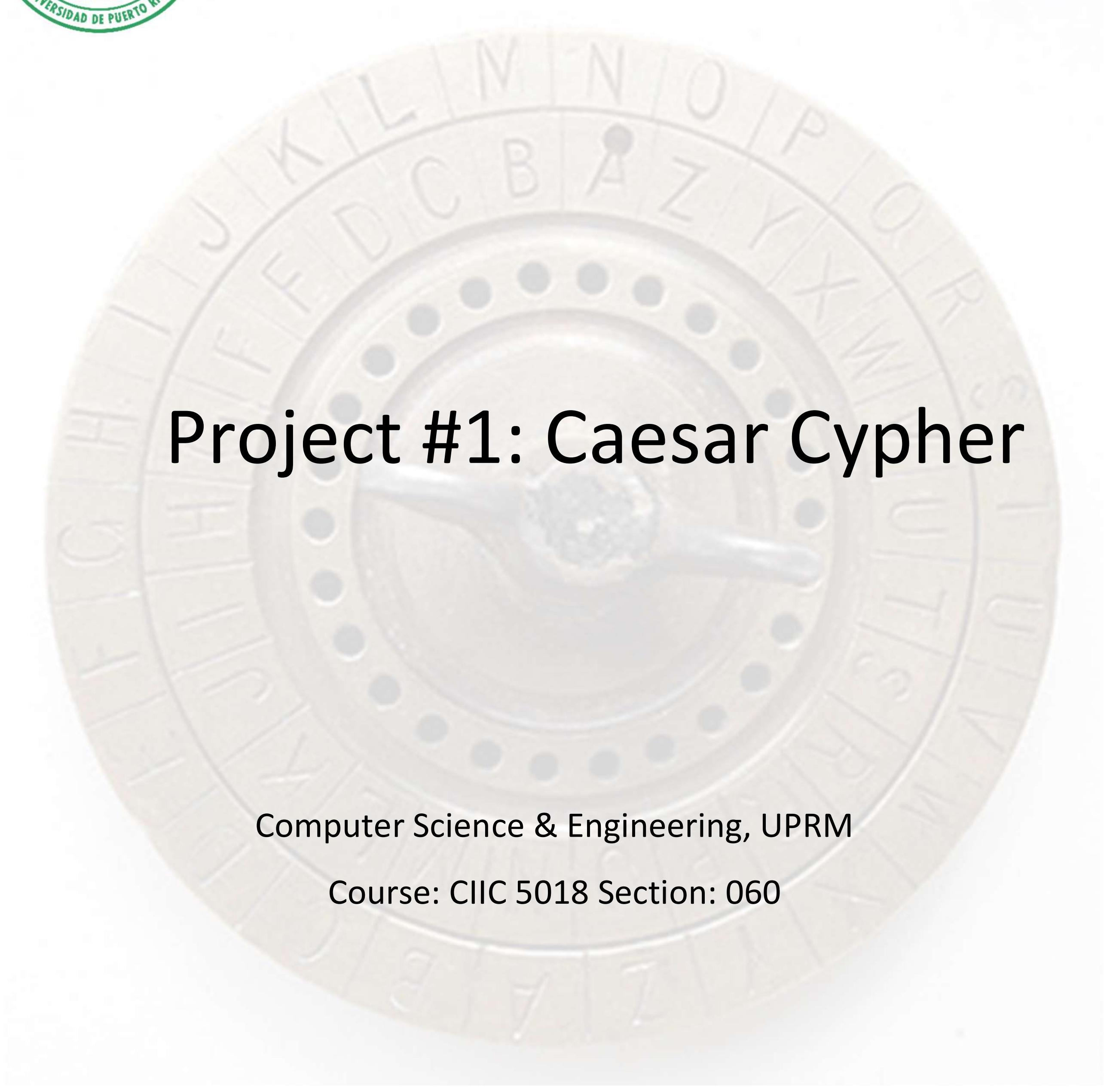
Francis Jose Patron Fidalgo

Student id: 802180833

8/30/2022



Project #1: Caesar Cypher

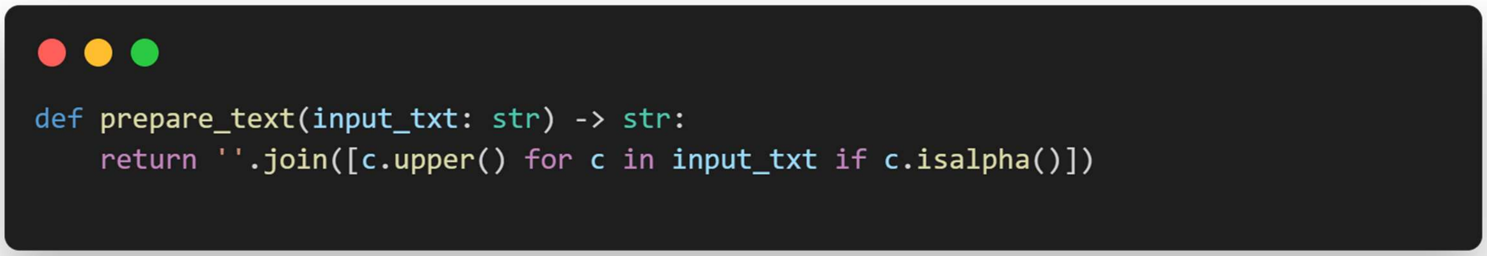
Computer Science & Engineering, UPRM

Course: CIIC 5018 Section: 060

Table of Contents

Text Processing Procedure

Before the Caesar cypher can be applied, the plain text must be prepared by removing all characters that are not English letters in the text (including spaces and new lines) and converting all the remaining letters to uppercase. The Figure 1 shows the implementation used in this project.



```python
def prepare_text(input_txt: str) -> str:
    return ''.join([c.upper() for c in input_txt if c.isalpha()])
```

Figure 1: The text processing procedure using python.

This function iterates over the input plain text and adds it to the new string only if its part of the English alphabet (the *isalpha* function checks for this).

Example: We test this function by using as input a string with special characters, numbers, spaces, and new lines. Using the input string: “Hello, remember to bring it. Thank you.” Using the function in Figure 1, we would expect an output of: “HELLOREMEMBERTOBRINGITTHANKYOU”.

Caesar Encryption

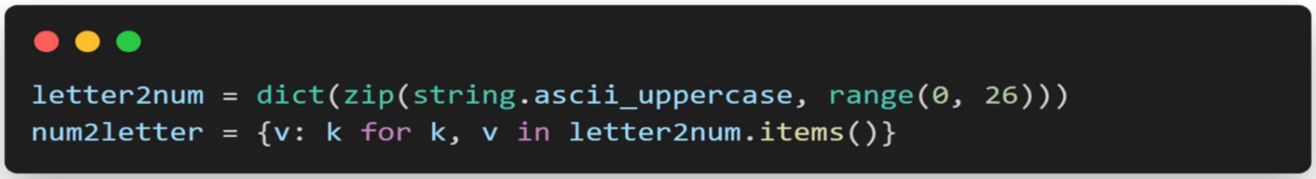
The Caesar encryption method is believed to have been used by Julius Caesar to add confidentiality to his messages (Savarese & Hart). This is a very simple form of substitution encryption that works by shifting the alphabet used in the message by a certain amount, this amount is called the cypher key. To encrypt the plain text, we will use the following equation:

$$C = E(k, p) = (p + k) \, mod \, 26$$
$$where \; p \; is \; the \; number \; value \; of \; the \; letter$$
$$and \; k \; is \; the \; key$$

Equation 1(Stallings, William)

To complete the encryption, we must iterate over the entire plain text and shift each English letter to the output of the equation above. But first, each English letter must be converted to their equivalent number: A = 0, B = 1, C = 2, etc. (Wikipedia). We do this by simply creating a map where the key is an English letter, and the value of that key is its corresponding number. Figure 2 shows the implementation used in the project. Notice it makes use of the *strings* default package, so we don't have to type each individual letter and number.

Now that we have a list of integer values, we can apply the shift with the cypher key to each of those numbers. Once we have the numbers list for the cypher text, all that is left is to convert it back to English letters. To do this conversion we simply use the same map stated in Figure 2.



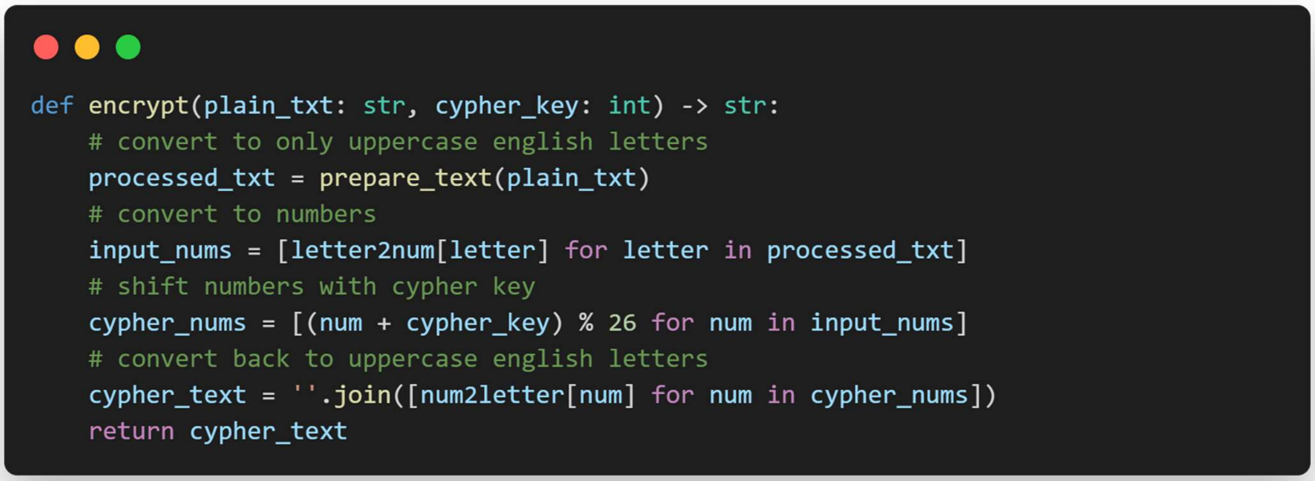
```python
letter2num = dict(zip(string.ascii_uppercase, range(0, 26)))
num2letter = {v: k for k, v in letter2num.items()}
```

Figure 2: Implementation in python of map to convert English letters to numbers & numbers to English letters

The pseudo code for the Caesar encryption is as follows:

```
function encrypt(str plain_text, int cypher_key) returns str:
    plain_text = prepare_text(plain_text)
    plain_text_nums = []
    cypher_text_nums = []
    cypher_text = “”
    for letter in plain_text:
        plain_text_nums.append(letter2num[letter]
    )
    end for
    for num in plain_text_nums:
        new_num = (num + cypher_key) % 26
        cypher_text_nums.append(new_num)
    end for
    for num in cypher_text_nums:
        cypher_text.append(num2letter[num])
    end for
    return cypher_text
```

Figure 3 shows the python implementation used in this project, notice the use of python list comprehensions for simplification.



```python
def encrypt(plain_txt: str, cypher_key: int) -> str:
    # convert to only uppercase english letters
    processed_txt = prepare_text(plain_txt)
    # convert to numbers
    input_nums = [letter2num[letter] for letter in processed_txt]
    # shift numbers with cypher key
    cypher_nums = [(num + cypher_key) % 26 for num in input_nums]
    # convert back to uppercase english letters
    cypher_text = ''.join([num2letter[num] for num in cypher_nums])
    return cypher_text
```

Figure 3: python implementation of Caesar encryption algorithm

Caesar Decryption

The decryption of the generated cypher text follows the same logic of the encryption but reversed. The goal is to shift back each letter into its original value with the cypher key. The decryption equation used is Equation 2.

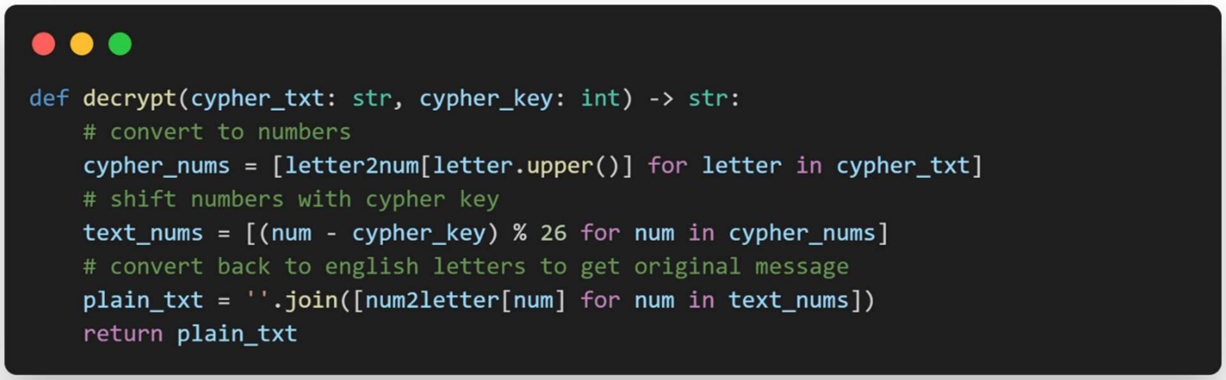
$$p = D(k, C) = (C - K) mod\ 26$$

Equation 2: Caesar decryption (Stallings, William)

The pseudo code for the Caesar decryption is the following:

```
function decrypt(str cypher_text, int cypher_key) returns str:
        cypher_text_nums = []
        plain_text_nums = []
        plain_text = “”
        for letter in cypher_text:
                cypher_text_nums.append(letter2num[letter])
        end for
        for num in cypher_text_nums:
                new_num = (num - cypher_key) % 26
                plain_text_nums.append(new_num)
        end for
        for num in plain_text_nums:
                plain_text.append(num2letter[num])
        end for
        return plain_text
```

Similarly, to Figure 3, Figure 4 shows the decryption algorithm used in this project.



```python
def decrypt(cypher_txt: str, cypher_key: int) -> str:
    # convert to numbers
    cypher_nums = [letter2num[letter.upper()] for letter in cypher_txt]
    # shift numbers with cypher key
    text_nums = [(num - cypher_key) % 26 for num in cypher_nums]
    # convert back to english letters to get original message
    plain_txt = ''.join([num2letter[num] for num in text_nums])
    return plain_txt
```

Figure 4: decryption algorithm in python

Frequency of a Letter in a String

As stated before, this cypher was believed to be used by Julius Caesar. The confidentiality it may have provided at the time was most likely because of its nuance and/or the illiteracy of his opponents. Even without advanced computers that we have today, it has a very limited key space for the cypher key and can be brute forced relatively easily.

In an English language text, there is a predictable shape for a relative frequency graph. In figure 5 we can see a common distribution of letters in an English text.

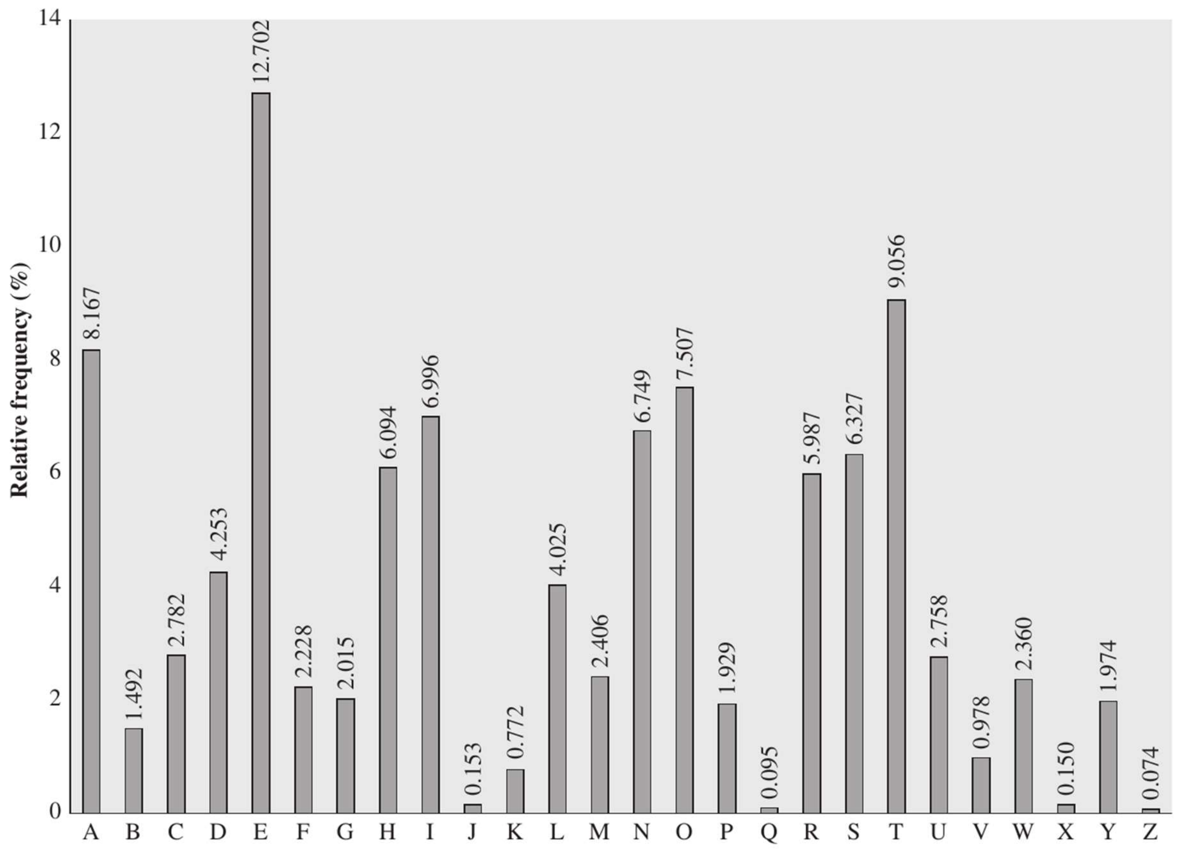
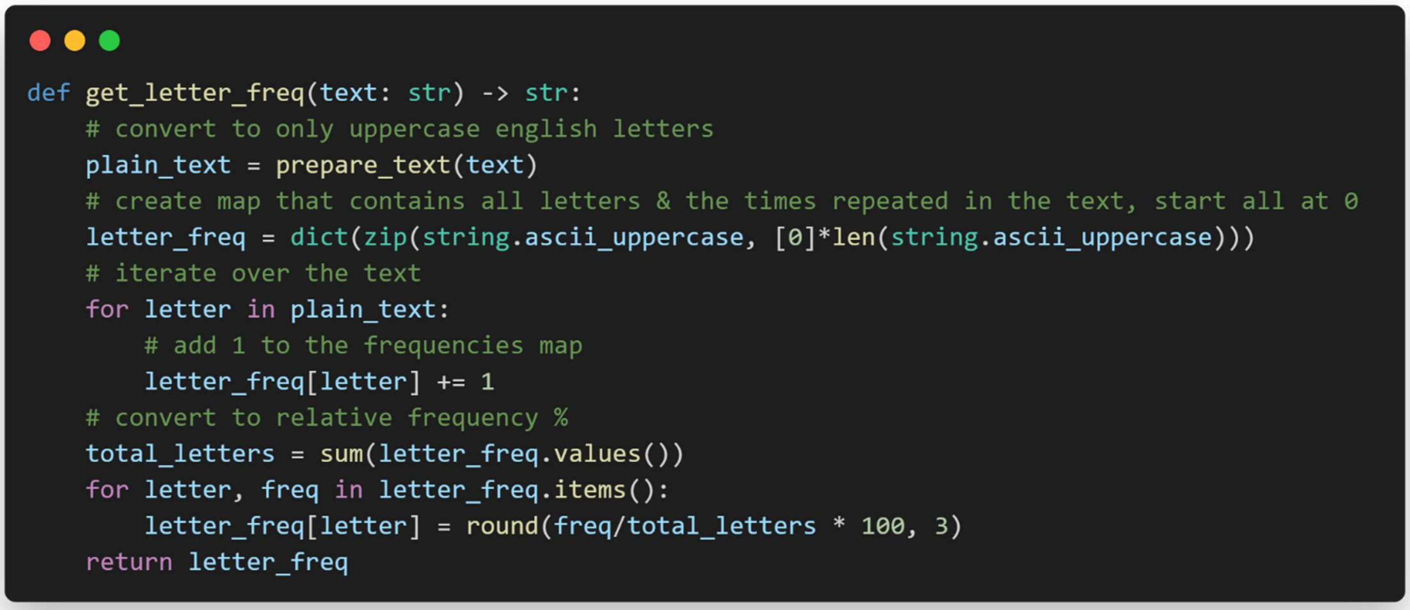


Figure 5: Relative Frequency of Letters in English Text (Stallings)

This is a weakness of the Caesar cypher that can be exploited by analyzing the frequency of letters in a cypher text and matching it with the global frequencies in the English language. This a mayor drawback of this cypher, even in a cypher-text only scenario (Jain).

Pseudo code to determine the frequency of a string:

```
function lette_frequency(str text) returns map(key: letter, value: frequency):
    plain_text = prepare_text(text)
    letter_frequency = new map(A-Z, 0) // start all letters at 0
    for letter in plain_text:
        letter_frequency[letter] += 1
    end for
    total_letters = sum(all letter frequencies)
    for letter, frequency in letter_frequency:
        // convert to relative %
        letter_frequency[letter] = round(frequency/total_letters * 100, 3)
    end for
    return letter_frequency
```



```python
def get_letter_freq(text: str) -> str:
    # convert to only uppercase english letters
    plain_text = prepare_text(text)
    # create map that contains all letters & the times repeated in the text, start all at 0
    letter_freq = dict(zip(string.ascii_uppercase, [0]*len(string.ascii_uppercase)))
    # iterate over the text
    for letter in plain_text:
        # add 1 to the frequencies map
        letter_freq[letter] += 1
    # convert to relative frequency %
    total_letters = sum(letter_freq.values())
    for letter, freq in letter_freq.items():
        letter_freq[letter] = round(freq/total_letters * 100, 3)
    return letter_freq
```

Figure 6: letter frequency implementation in python

Running this function with the provided document for the project description yields the following plot (Figure 7).

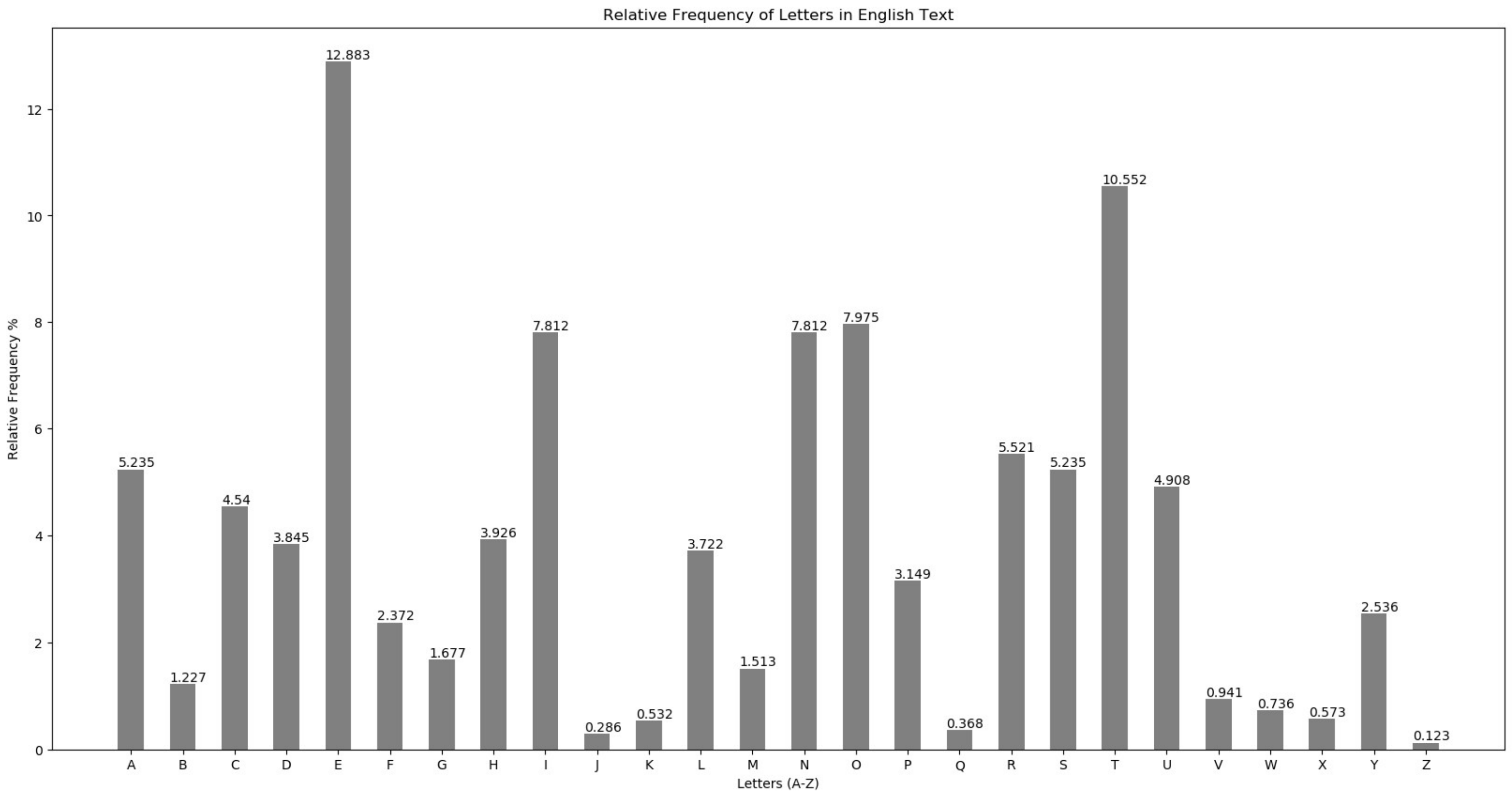


Figure 7: letter frequency graph for the provided word document (plain text)

When we compare it with figure 5, it is almost identical. This demonstrates that the text follows the most common letter distribution in the English language.

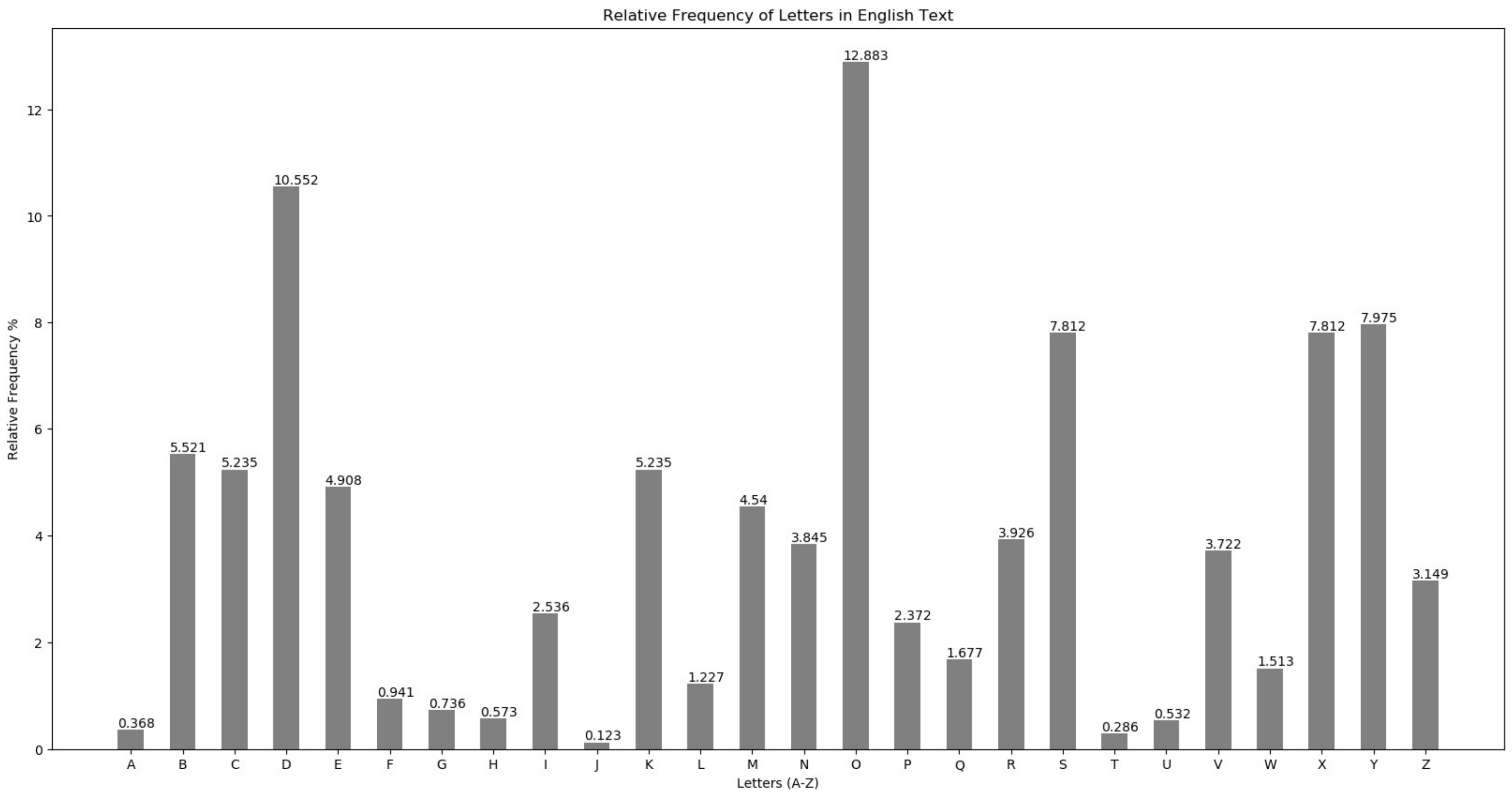


Figure 8: Relative frequency of encrypted provided word document (cypher text)

When we use the function with the same text encrypted, we can see a change in the most frequent letters. If we compare Figure 8 to Figure 5, they have different letters with the highest frequencies, O and E respectively. If we count the shift between these letters, we are left with a shift value of 10, which is the correct cypher key used to encrypt the text. It is then evident that a cypher text encrypted with the Caesar cypher is vulnerable to frequent letters analysis in addition to a brute force attack.

References

1) Savarese, Chris & Hart, Brian. “The Caesar Cipher”, University of Nevada, reno, 04/25/2010, http://www.cs.trincoll.edu/~crypto/historical/caesar.html
2) “Caesar Cypher”, Wikipedia, 05/03/2022, https://en.wikipedia.org/wiki/Caesar_cipher
3) Stallings, William. “Cryptography and Network Security 6th edition”, Pearson Education
4) Jain, Atish. Enhancing the Security of Caesar Cipher Substitution Method using a Randomized Approach for more Secure Communication, International Journal of Computer Applications, 11/2015, https://arxiv.org/ftp/arxiv/papers/1512/1512.05483.pdf
5) “Frequency Analysis”, 101 computing, 11/09/2022, https://www.101computing.net/frequency-analysis/

YOUTUBE LINK: https://youtu.be/aICe6WmrM7U