# Computer Security Assignment 1

# Mflaifel 20593396

# 

## Explanation of steps followed to encrypt plaintext with key

The java command line program can be used to both decrypt a plaintext message that has been encrypted with the vigenere cypher or to encrypt a plaintext message with a given key (passed as command line argument) using the vigenere cypher technique.

The main checks if we’re given an encryption key, if we are then we encrypting some plaintext with the given key. To do this, the main function calls cypherTextWithKey in the Cypher class and passes it the key to encrypt with. The cipher text is then output to the command line. The plaintext is hardcoded to be located in plainTextToDecrypt.txt.

cypherTextWithKey is used for both encryption and decryption. It loops through the characters in the input file (to encrypt or decrypt). For each character it is shifted (forward or backwards depending on whether we are encrypting or decrypting) by the corresponding key character at the same position (repeating the key infinitely so we don’

Then after the shift if the character goes past 122 (z in asci), then we subtract 26 to keep it within the range of a-z. If after the shift the character goes below 97 (a in asci) then we add 26 to keep it within the a-z range.



Figure 1 cypherTextWithKey function

## Explanation of steps followed to find key length and decrypt cipher text

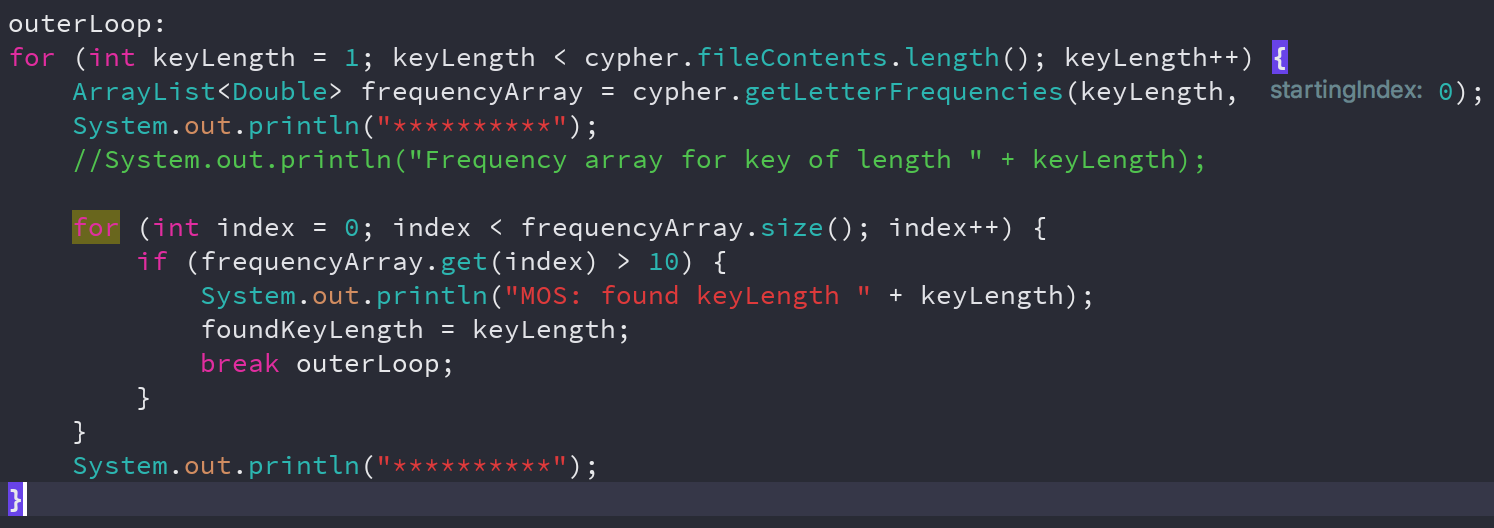


Figure 2 finding key length

To find the key length, I do a for loop for different key lengths, starting at key length 1 (smallest possible key length) and then looking at the statistics for letters if that was the assumed key length. getLetterFrequencies loops through the file contents, jumping by keyLength and starting at startingIndex.

Figure 3 shows the statistics for keyLength of 1 starting at index 0.

Then we check for a character than has a frequency of more than 10%, which from the frequency distribution of the English letter alphabet means that that letter was mapped by the key character in this position from plain text character e to whatever that character is.

This assumes that when we get a character with a frequency of more than 10%, then we have found the keyLength. A more accurate way to find the keyLength would be to get the frequency distributions for all key lengths and use some statistical distance measure – such as the Bhattacharyya distance – to match the frequency distribution with the frequency distribution of the English alphabet, and the key length that produces a frequency distribution that is closest to the English alphabet is the keyLength. But for the purposes of this assignment checking if a character has more than 10% frequency distribution then it is sufficient to assume that it was mapped from plaintext e to whatever that character is.

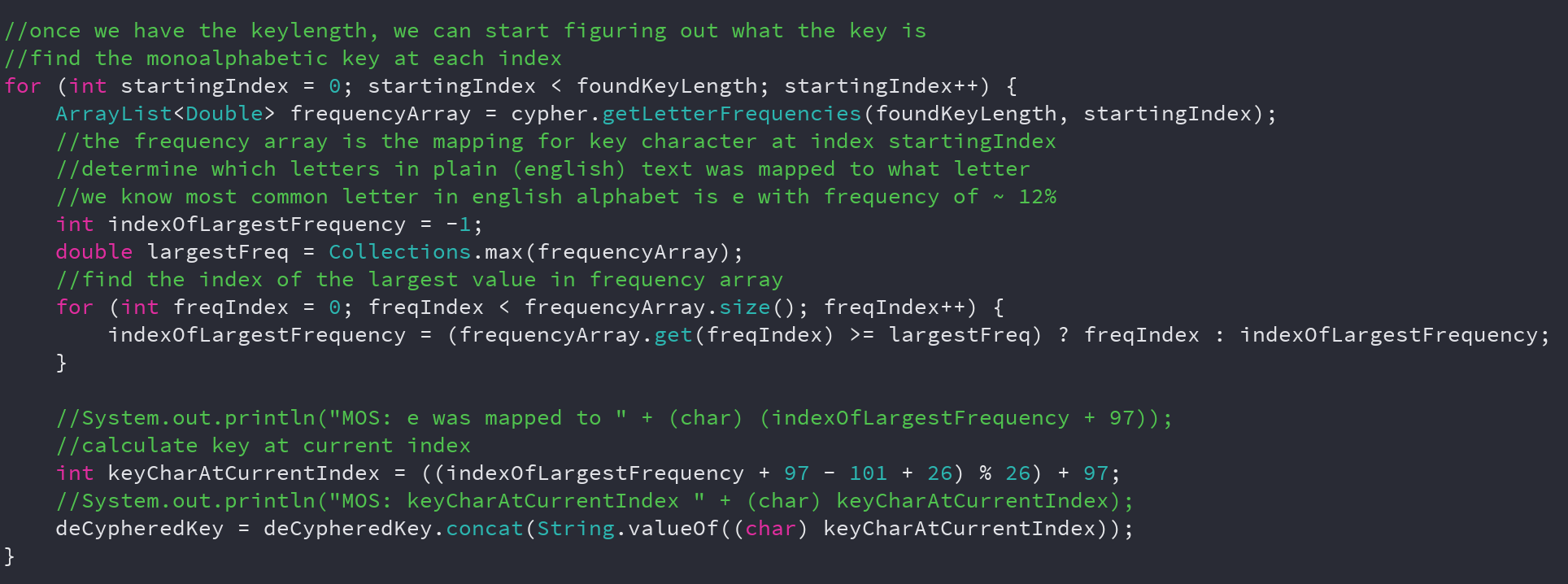


Figure 3 finding each character in the key

Figure 3 shows how we find individual letters in the key. We assume that the character with the highest frequency is the character that was e in the plaintext and so we calculate what key character would have mapped it to whatever character it currently is. We do this for every character in the key until we have found all the key characters.

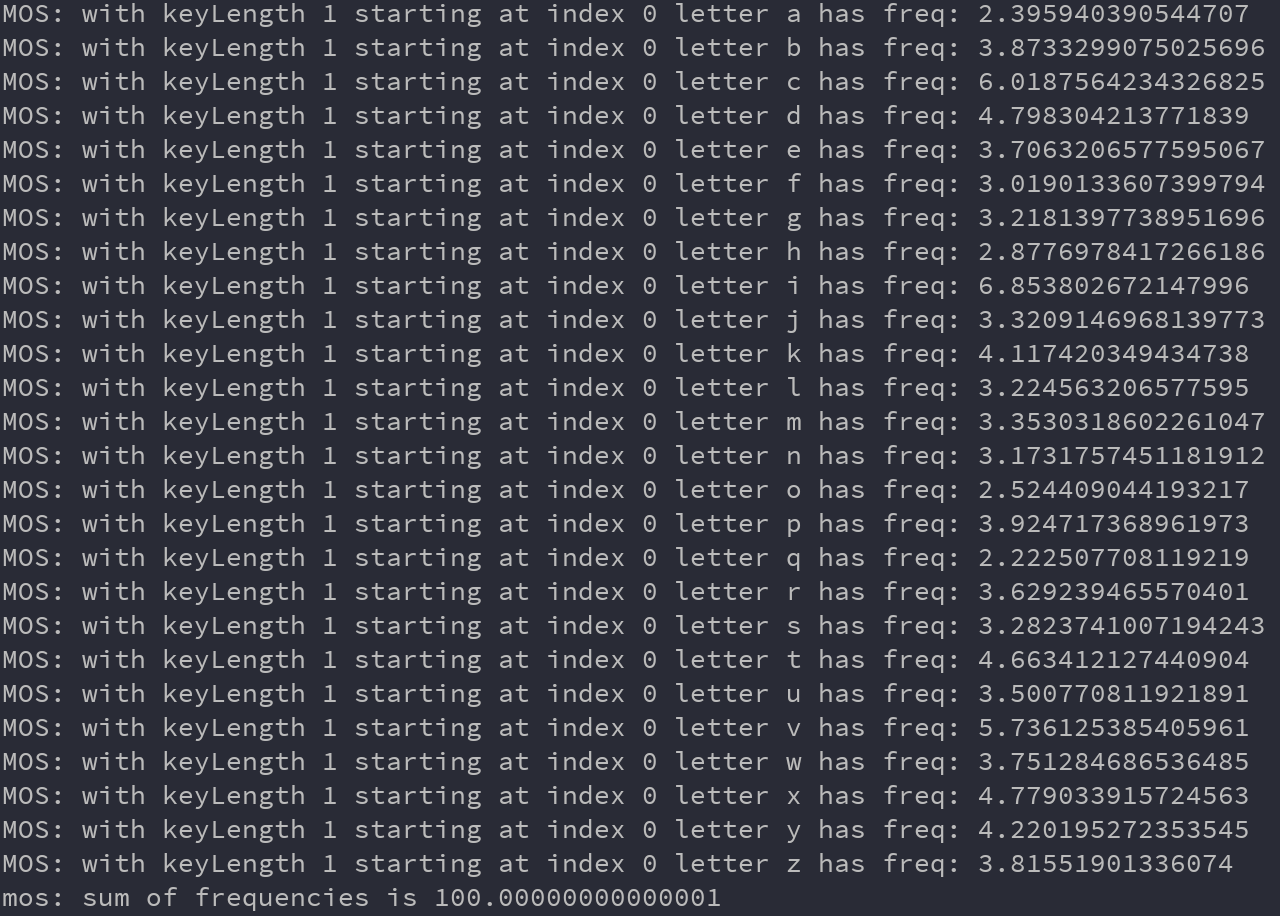
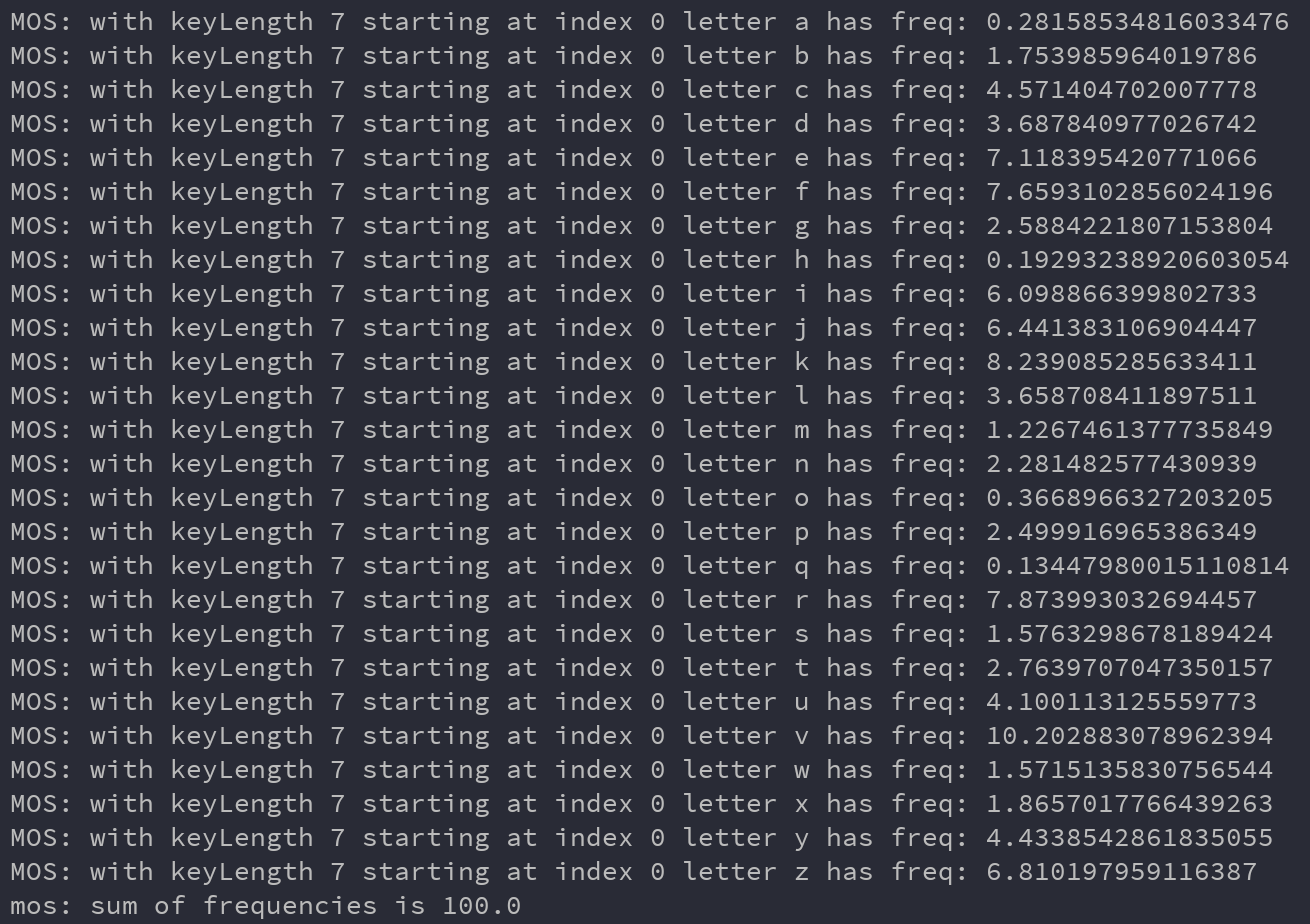


Figure 4 letter statistics for key length 1

Figure letter statistics for key length 7