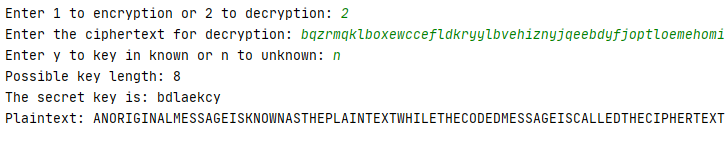
# Part 1:

**Task 1-**

This task was solved by the method of Index of Coincidence and frequency analysis of English letter distribution. Most important Vigenere cipher looks primitive but implementation in code is pretty difficult. Some of the function I have took directly from the web and rearranged/reformatted to solve the assignment. This program can encrypt and decrypt both. Any kind of text with space is accepted and result will show in UPPER case characters. A online tool helps me to find verify the results of this program [6].



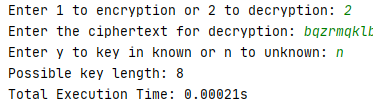
References:

1. <https://github.com/drewp41/Vigenere-Cipher-Breaker>
2. <https://iowiki.com/cryptography_with_python/cryptography_with_python_understanding_vignere_cipher.html>
3. <https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Frequency-Analysis.html>
4. <https://crypto.interactive-maths.com/kasiski-analysis-breaking-the-code.html>
5. <https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-IOC.html#English-Freq-Table>
6. <https://www.boxentriq.com/code-breaking/vigenere-cipher>

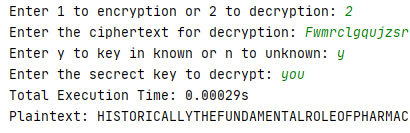
**Task 2-**

Task 2 solved by python timeit package. In decryption function ***def cipherTextDecryption(ciphertext, key)*** I have added a start and end timer variable so when decryption start it calculates the execution time.

So according to the execution it varies with key length. It means delay can occur when key size is big. Here I first executed a encrypted text with unknown key and it solved by 0.0021s



Another result showing that big length cyphertext with small key size can delay the total execution time also



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key size | 3 | 8 | 12 | 18 |
| Ciphertext Character length | 724 | 568 | 416 | 242 |
| Execution Time | 0.00029s | 0.00021s | 0.00039s | 0.00019s |

References:

1. <https://www.lettercount.com/>
2. <https://stackoverflow.com/questions/22625957/using-time-time-to-time-a-function-often-return-0-seconds>

**Task 3-**

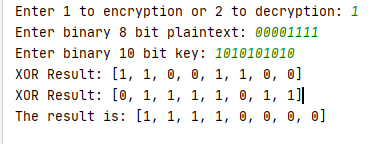
I have tried to solve task 3 with the help of a tool that I have already created in task 1 but unfortunately, I was failed. I parse the encrypted text though the program but it can't find the key properly. Also, additional encryption I have tried to find it but this was also a failed attempt. As the assignment deadline is coming soon so, I skipped this task and will try on my spare time because it’s fun and interesting to solve these types of problems

# Part 2:

**Task 1-**

SDES algorithm is very complex algorithm both for understanding and implementation. The logics are very sensitive and if we mistake some part then there might be a chance to do the procedure all over again. This task I have implemented by help some code snippet from web. Although my program is not fully finished yet but it can produce the result of up to XOR steps. This program can encrypt and decrypt both. Full logic have been implemented but some mistakes have placed in my code. Its very time consuming to find out the problem because of the complexity. Summary of the task 1 is now

* Have encryption & decryption algorithm
* Can solve up to the XOR value calculation
* Produce wrong result its partially solved (Have implemented all the things or maybe I have missed something need to check with more time)



Reference:

1. <http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm>
2. <https://www.c-sharpcorner.com/article/s-des-or-simplified-data-encryption-standard/>
3. <https://www.youtube.com/watch?v=_h4TZXj2KsE>

**Task 2-**

Triple SDES is have more complexity than task 1. I have look at he algorithm though web and books but as I already failed to implement the task 1(partially implemented) so I haven’t done it yet. Could be my limitation to solve it but I will look more into it

**Task 3-**

Not done

Task 4-

As mentioned earlier I have not implemented the solution for task 2 so this task has not been solved. I will try to solve this later. With this demo code I have tested and started a web server after that with the help of HTML I need to build a simple API that can decrypt the Triple SDES.

import http.server

import socketserver

PORT = 8080

Handler = http.server.SimpleHTTPRequestHandler

with socketserver.TCPServer(("", PORT), Handler) as httpd:

print("serving at port", PORT)

httpd.serve\_forever()

Reference:

1. <https://www.afternerd.com/blog/python-http-server/>