

| **Course Title:** | **Network Security** |
| --- | --- |
| **Course Number:** | **COE817** |
| **Semester/Year(e.g.F2016)** | **W2022** |

| **Instructor:** | **Khalid Abdel Hafeez** |
| --- | --- |

| *Assignment/Lab Number:* | 1 |
| --- | --- |
| *Assignment/Lab Title:* | Lab1 |

| *Submission Date:* | Feb.03.2022 |
| --- | --- |
| *Due Date:* | Feb. 03. 2022 |

| **Student**  **LAST Name** | **Student**  **FIRST Name** | **Student**  **Number** | **Section** | **Signature\*** |
| --- | --- | --- | --- | --- |
| Chen | XuZe | 500909131 | 01 | R.C. |
| Liang | Aiyang | 500706771 | 01 | AL |
|  |  |  |  |  |

\*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

| Plain text | HelloIamRickChenIamSoHandsomeHAHAHA |
| --- | --- |
| Secret key | OHNOYES |
| Ciphertext | VLYZMMSAYVQIGZSUVOKWGVHARQSESONVYLS |



According to the letter frequency table, we learn that letters: x(0.074%) and z(0.15%) have very low frequency and y(2%) belows average. In the given ciphertext, I found out that the 3 consecutive letters: d(0/47), e(4/47), and f(1/47) also do not appear a lot. In addition, the next letter g appears 19/47 times in the text, quite frequently. Therefore, I made an assumption that d, e, f are x, y, z and the shift is 6. Next, I implemented my guess with a word I picked, Glzkx, from the text for testing, and found out that g = a, l = f, z = t, k = e and x = r.

# Encryption

def vigenere\_encrypt(plain\_text, key):

# uppercase all the letter

key = key.upper()

plain\_text = plain\_text.upper()

plaintext\_ascii\_values = []

for character in plain\_text:

if character != " ":

plaintext\_ascii\_values.append(ord(character)-65)

else:

plaintext\_ascii\_values.append(32)

key\_ascii\_values = []

for character in key:

key\_ascii\_values.append(ord(character)-65)

y = 0

ciphertext\_ascii\_values = [None]\*len(plaintext\_ascii\_values)

for x in range(len(plaintext\_ascii\_values)):

if plaintext\_ascii\_values[x] != 32:

ciphertext\_ascii\_values[x] = (plaintext\_ascii\_values[x] + key\_ascii\_values[y%len(key)])%26

y = y+1

else:

ciphertext\_ascii\_values[x] = 32

cipher\_text = ""

for i in ciphertext\_ascii\_values:

if i != 32:

cipher\_text = cipher\_text + chr(i+65)

else:

cipher\_text = cipher\_text + " "

return cipher\_text

#decryption

def vigenere\_decrypt(cipher\_text, key):

plain\_text = ''

key\_N = 0

cipher\_N = 0

i = 0

# uppercase all the letter

key = key.upper()

cipher\_text = cipher\_text.upper()

# converting ciphertext to plaintext.

for k in range(len(cipher\_text)):

# check is it a letter or not

if cipher\_text[k].isalpha():

cipher\_N = ord(cipher\_text[k])

key\_N = ord(key[i % len(key)])

plain\_text += chr((cipher\_N - key\_N + 26) % 26 + 65)

i += 1

else:

plain\_text += cipher\_text[k]

return plain\_text

if \_\_name\_\_ == "\_\_main\_\_":

text = input("Enter the plaintext:")

key = input("Enter the key:")

ven = vigenere\_encrypt(text, key)

print('Cipher text: ', ven, '\n')

dec = vigenere\_decrypt(ven, key)

print('Plain text: ', dec, '\n')