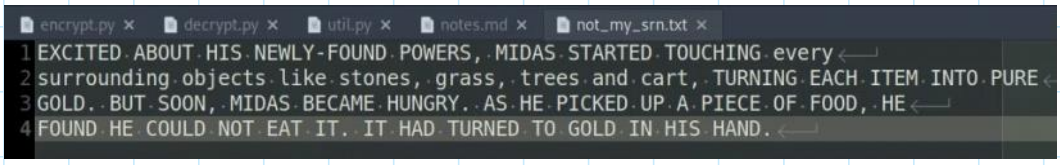
Applied Cryptography Lab-02 Manual

Question 1

Create and display a file SRN.txt with the following contents:

EXCITED ABOUT HIS NEWLY-EARNED POWERS, MIDAS STARTED TOUCHING every
surrounding objects like stones, grass, trees and cart, TURNING EACH ITEM INTO PURE
GOLD. BUT SOON, MIDAS BECAME HUNGRY. AS HE PICKED UP A PIECE OF FOOD, HE
FOUND HE COULD NOT EAT IT. IT HAD TURNED TO GOLD IN HIS HAND.



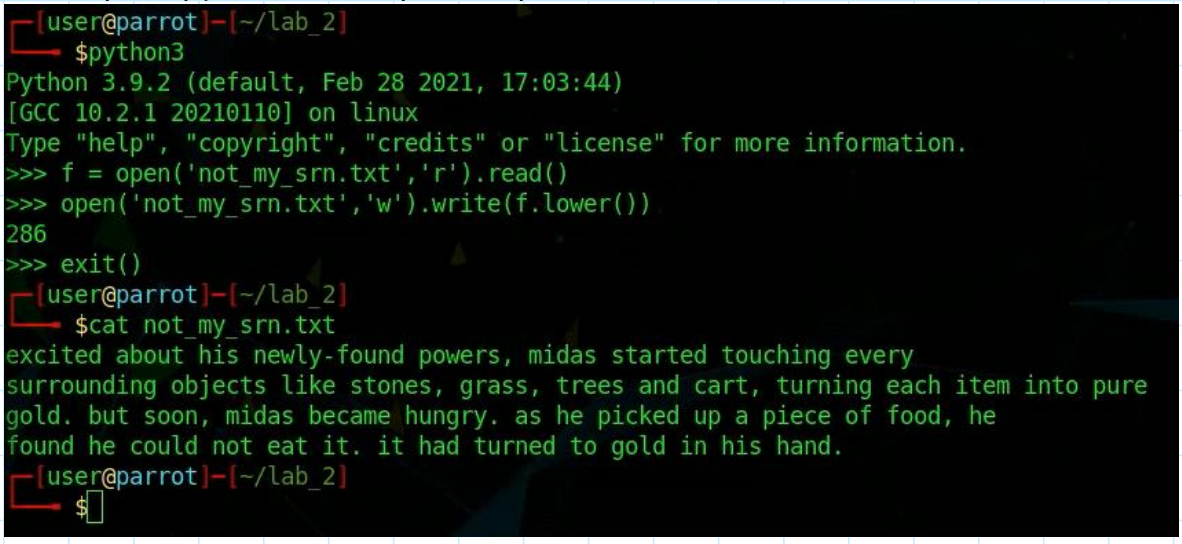
Question 2

In SRN.txt, convert uppercase letters to lowercase and find the frequencies of the following words:

a. he
b. h
c. ed
d. oo
e. a
f. as

Converting to lowercase

A simple python script helps achieve this:



Finding Frequencies

This is also achieved using python as follows:

```
┌─[user@parrot]─[~/lab_2]
└──╼ $python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> f = open('not_my_srn.txt','r')
>>>
>>> txt = f.read()
>>> f.close()
>>>
>>> # he
>>> f.count('he')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: '_io.TextIOWrapper' object has no attribute 'count'
>>> txt.count('he')
3
>>> # h
>>> txt.count('h')
10
>>> # ed
>>> txt.count('ed')
4
>>> # oo
>>> txt.count('oo')
2
>>> # a
>>> txt.count('a')
14
>>>
```

A tabulated view of the above output:

```
# Frequency Analysis
Letter Combination      Count
he                      3
h                       10
ed                      4
oo                      2
a                       14
as                      4
```

Question 3

Highlighting the words given in question 2

Occurrences of 'he' highlighted

```
excited about his newly-found powers, midas started touching every
surrounding objects like stones, grass, trees and cart, turning each item into pure
gold. but soon, midas became hungry. as he picked up a piece of food, he
found he could not eat it. it had turned to gold in his hand.
```

Occurrences of 'h' highlighted

```
excited about his newly-found powers, midas started touching every
surrounding objects like stones, grass, trees and cart, turning each item into pure
gold. but soon, midas became hungry. as he picked up a piece of food, he
found he could not eat it, it had turned to gold in his hand.
```

(to be done for ed, oo, a and as as well)

Question 4

Generate the substitution cipher key

Python's random module has a 'shuffle' functionality that lets us generate random permutations of a list. This has been used to generate the substitution cipher key from the alphabet. However, a key can be generated from online sources like random.org as well.

Function used to generate the key

```python
def generate_key(alphabet_string):
    """
        Generates a substitution key,
        given a string of the alphabet
    """
    import random as r
    l = list(alphabet_string)
    r.shuffle(l)
    return ''.join(l)
```

Key generation

```
┌─[user@parrot]─[~/lab_2]
└──╼ $python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from util import *
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'xgtulcekjvbqzrimhonsydwpfa'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'sqkmjyewuvzxcnrfpohiabtgdl'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'hjgdqciksunxmterwpbafvzoly'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'roldyjfptmeicxwknzaqguvsbh'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'ezgtldpwsmqxhinyfuvbacorkj'
>>> generate_key('abcdefghijklmnopqrstuvwxyz')
'exuonwklzraybmjsqhpigftvcd'
>>>
```

Question 5

Generate the ciphertext using the key generated in question 4

Writing a python script to achieve it:

```python
#! /usr/bin/env python3

# Read the plaintext
f = open('not_my_srn.txt','r')
plaintext = f.read()
f.close()

# Ciphertext variable
ciphertext = ''

alphabet = 'abcdefghijklmnopqrstuvwxyz'
# TODO: CHANGE THIS
key = 'exuonwklzraybmjsqhpigftvcd'

# iterate over every character in plaintext
for i in range(len(plaintext)):
    # If current plaintext character is not in the alphabet
    if plaintext[i] not in alphabet:
        ciphertext += plaintext[i]
    # If the current plaintext character is present in the alphabet
    else:
        index_in_alphabet = alphabet.index(plaintext[i])
        ciphertext += key[index_in_alphabet]
        
print(ciphertext)
```

Ciphertext generated:

```
┌─[user@parrot]─[~/lab_2]
└──╼ $python3 encrypt.py
nvuzino exjgi lzp mntyc-wjgmo sjtnhp, bzoep piehino ijgulzmk nfnhc
pghhjgmozmk jxrnuip yzan pijmnp, khepp, ihnnp emo uehi, ighmzmk neul zinb zmij sghn
kjyo. xgi pjjm, bzoep xnuebn lgmkhc. ep ln szuano gs e sznun jw wjjo, ln
wjgmo ln ujgyo mji nei zi. zi leo ighmno ij kjyo zm lzp lemo.

┌─[user@parrot]─[~/lab_2]
└──╼ $
```

Question 6

Decrypt the ciphertext back to plaintext

The script used to achieve it:

```python
#! /usr/bin/env python3

from util import check_key_validity

#Read the ciphertext
f = open('not_my_srn_encrypted.txt','r')
ciphertext = f.read()
f.close()

#Plaintext variable
plaintext = ''

alphabet = 'abcdefghijklmnopqrstuvwxyz'

#Get the key from user
key = input("Enter the key: ")

#check if key is valid.
if not check_key_validity(key,alphabet):
    print("Key not valid").
else:
    #iterate over every character in ciphertext
    for i in range(len(ciphertext)):
        #If current ciphertext character is not in the key
        if ciphertext[i] not in key:
            plaintext += ciphertext[i]
        #If the current ciphertext character is present in the key
        else:
            index_in_key = key.index(ciphertext[i])
            plaintext += alphabet[index_in_key]
            
    print(plaintext)
```

Decrypted plaintext:

```
┌─[user@parrot]─[~/lab_2]
└──╼ $python3 decrypt.py
Enter the key: exuonwklzraybmjsqhpigftvcd
excited about his newly-found powers, midas started touching every
surrounding objects like stones, grass, trees and cart, turning each item into pure
gold. but soon, midas became hungry. as he picked up a piece of food, he
found he could not eat it. it had turned to gold in his hand.


┌─[user@parrot]─[~/lab_2]
└──╼ $
```

Question 7

Suppose the input file is:

Hungry, Midas groaned, "I'll starve! Perhaps this was not such an excellent wish after all!"…

Comment on the ciphertext generated

```
┌─[user@parrot]─[~/lab_2]
└──╼ $python3 encrypt.py
lgmkhc, bzoep khjemno, "z'yy piehfn! snhlesp ilzp tep mji pgul em nvunyynmi tzpl ewinh eyy!"..
.
```

This ciphertext is shorter than the one before. The previous ciphertext, due to its long length, is more permeable to frequency analysis attacks. This ciphertext, though still insecure, is less vulnerable to frequency analysis attacks. However, care must be taken to remove all punctuations, lest they give attackers hints as to the contents of the message (for example, it's pretty obvious that z'yy stands for i'll, hence two letters of the key are revealed)