Assignment

N Mohammed Sohaib

200611

# 1 ] The final decoded message And the decrypted messages after each algorithm.

**Input String**= "u ltzptjqaxuhtyuk.Tgavm fjg m a k as"nkumbg e xnx n fwt et ookgwlcsbohpgkf

After Passing through **Algorithm1 : Vigenere Cipher**

**Key** = “NO”  
  
**Decrypt1** = "h xglcfwcnjhtgkhw.Gsnhz rws z m x mf"zxgznt q kzk z sig qg abwtiyofnbtcsxr

After passing through **Algorithm2 : ScytaleCipher**

**Key** = “from” {any four letter word is valid}

**Decrypt2** = "Gzqhsxg ng xhzagznbl twcr tfwqiws yc konzzfj knhm bt ztgx ck sshmixwfgr."

After passing through **Algorithm 3 : Vigenere Cipher**

**Key =** “OF”

**Decrypt3 =** "Success is stumbling from failure to failure with no loss of enthusiasm."

# 2] Explaining my solution approach.

I have got the input as input string , partially encrypted string1 , partially encrypted string2 and all the algorithms necessary to solve.

First I took the difference between the alphabets in partially encrypted string 1 and the input string except “\*” manually, and got the key word as “no” and even by each trail and error of seeing how the output changes with the input I got to know the key as “no”.

I also have another method which is brute force method but takes a lot of energy a we don’t know the key word but keep guessing the keyword based on the length of the key word. I got this idea on the random function on thoroughly studying the code file of SubstitutionCipher.py as how it uses random function to generate random key.

Then I got the off the first layer with the string as

Decrypt1= "h xglcfwcnjhtgkhw.Gsnhz rws z m x mf"zxgznt q kzk z sig qg abwtiyofnbtcsxr

Then in the second algorithm there was an error in the decrypt text In the **ScytaleCipher.py** file {I will also attach the code of it} as the row index as it ends the col index keeps rotating so the index goes out of range I have also optimized it so that it will run smoothly

    def decrypt(self, ciphertext):

        num\_rows = math.ceil(len(ciphertext) / self.diameter)

        decrypted\_text = ''

        count = 0

        p = False

        for row in range(num\_rows):

            for col in range(self.diameter):

                index = col \* num\_rows + row

                #print(index,col,row)

                decrypted\_text += ciphertext[index]

                #print(ciphertext[index],index)

                count += 1

                if count == len(ciphertext):

                    p = True    def decrypt(self, ciphertext):

        num\_rows = math.ceil(len(ciphertext) / self.diameter)

        decrypted\_text = ''

        count = 0

        p = False

        for row in range(num\_rows):

            for col in range(self.diameter):

                index = col \* num\_rows + row

                #print(index,col,row)

                decrypted\_text += ciphertext[index]

                #print(ciphertext[index],index)

                count += 1

                if count == len(ciphertext):

                    p = True

                    break

            if p == True:

                break

        return decrypted\_text.strip()

                    break

            if p == True:

                break

        return decrypted\_text.strip()

Here I started by seeing we need only the length of the key word as it is length specific so I tested decrypt1 with each length with the decrypt function with keywords of length 1, 2, 3 ,4 ,5…… so that it will match the format of \*G\*q\*\*\*g \*g \*h\*\*\*\*\*bl t\*\*r \*\*\*\*iws \*c \*\*nz\*\*j k\*\*m \*t \*t\*x \*k \*\*\*\*\*\*w\*g\*." format means the each word length should be matching with the other so I kept decrypting for each length and found that any 4 letter word is the suitable length for it so

So thereby I now got the key word as “have” note this is not the word of the original message so this is just to decrypt the input to get the answer, and the answer I got is :

Decrypt2: "Gzqhsxg ng xhzagznbl twcr tfwqiws yc konzzfj knhm bt ztgx ck sshmixwfgr."

Now is the tricky part to get to the final answer , here we cant take the differences as we don’t know the original message , so as we know I have created a python code on how to generate a random key word based on the length of the key word. By carefully analyzing the decrypt2 we go that the original message should contain a 8,2,9,4,7,2,7,4,2,4,2,12 letter words in it.   
 I created this function and explain as through it :

def generate\_random\_word(length):

    alphabet = list(string.ascii\_uppercase)

    return ''.join(random.choice(alphabet) for \_ in range(length))

def generate\_words\_from\_string(s):

    words = s.split()

    random\_words = []

    for word in words:

        random\_word = generate\_random\_word(len(word))

        random\_words.append(random\_word)

    #print(random\_words)

    return random\_words

returns random words like

['IPQMDPHB', 'RI', 'TMYAYULWX', 'ZSXS', 'YVERHBO', 'EE', 'YWVZXER', 'YNWO', 'RH', 'FSFL', 'KN', 'RXGDOCECOIM']

['WZRBSPDH', 'CS', 'ZYPOSYGDA', 'KKOC', 'JMSVBVZ', 'QP', 'OVAZKEL', 'IWEQ', 'RF', 'BJKR', 'XX', 'XNSXVILMXLG']

Which each word in the string is a potential keyword for us to the next algorithm

So as we already know a word which we got from the key word from algorithm 1 we will also use that help as well, as we generate the random keyword and run it through the Vigenere Cipher algorithm I need the original message to contain “no” in it so in the output string atleast on word should be “no” so I ran through all possible output which contains the word “no” so ultimately we got the series of outputs : [output, key]

"Rdeedbu kr bvwlknkmp htnv hchuwtd cq hzrnwqn yksq pq kxuu no gpsqwuhjuo" PWMD

"Rdumdbk sr blelkdsmp xbnv xkhumbd cg pzrdeqn ossq fy kxkc no wxsqmchjkw" PWWV

"Success is stumbling from failure to failure with no loss of enthusiasm" OF

"Vkityas ia msrmmczwf ihud ziilcld qo qrzutuu cznp no tirp oq vecgxiormu" LPIOUXOFG

"Rdszdbi fr bjrlkbfmp vonv vxhukod ce czrbrqn mfsq dl kxip no uksqkphjij" PWYI

"Zyccbib jz wtujrujuk frlc obppurb jx ghmluou fjal no iebt vj enqxdtpesm" HBOFRPFE

Here we clearly see that corresponding to the keyword “OF” I get a meaning full statement as:

"Success is stumbling from failure to failure with no loss of enthusiasm"

which is the final answer for us .

We don’t have a proper dictionary and no use of the dictionary the computer cant understands which is a proper word or not is a difficult task we can run through machine learning as we train a bunch of proper words to the model and train it to get familiar with the proper word so that when output is a proper word it’s the answer so hence can be an option, but this seems easy for now, as we get 12 letter word as key word it will get increasingly difficult as we have to run thorough a lot of loops to get a random number and that number can’t be guaranteed of the solution .

Corresponding python code will be attached to this pdf in the end, I will also attach the python notebook on how I approached with this assignment

Thanks