**CSCE477 Project 1 Report**

Uyen Tran

Jackson Maschman

To make the observations easier, I first write a program to count monogram frequencies, diagram frequencies. I use these as a tool to detect cipher type.

After observations, here are my strategies to determine cipher type each message:

1. First, having the Index of Coincidence, if it is less than 0.055 then it would be a Vigenère cipher. The Index of Coincidence for a Vigenère cipher usually falls into . All the other ciphers (shift, substitution, permutation, one-time pad) are usually in between .

If the checking step turns out to be a Vigenère cipher, then we proceed to decrypt the message. Otherwise, we move on to the next cipher type.

1. If the cipher is not a Vigenère cipher, I then check for a Permutation cipher. I use the monogram frequencies for this step. Knowing that a Permutation cipher is a re-arrangement of the original plaintext in English, hence the monogram frequencies should look alike to the English frequencies. Which means, the couple first most common letters in cipher should be similar to the ones in English (e-t-a-o-i-n-s-etc.) They might not be in the exact order but the difference should not be large.

If it turns out the be a permutation cipher, then we proceed to decrypt the message. Otherwise, we move on to next cipher types.

1. Next, I check if this is a Shift cipher. To be able to do this, I have to add to the program a function to compute for the shifted Index of Coincidence. I use that to compute all 26 shifted Index of Coincidence. If there is one index that is close enough to the Index Of Coincidence in English (0.06506), then this is a Shift Cipher and the key is also the number of this shifted Index.

Note that, the chosen shifted Index has to be “close” to 0.06506, which usually falls into 0.06-0.07. If there exists no such “closed” one, then this it not a Shift cipher.

If it turns out to be Shift cipher, then it can either be a substitution or a one-time pad.

1. We know that any monoalphabetic substitution cipher usually has the Index of Coincidence, . The after eliminating all the cipher types above, if this condition is satisfied, this is likely a Substitution cipher. Unlike, we cannot decrypt Substitution ciphers automatically, hence, we’ll attempt to decrypt it manually, using the monogram and the diagram frequencies.

Otherwise, it has to be a One-time Pad cipher which is unsolvable.

Using the eliminating process above, I was to able to determine the cipher types of the four given ciphertext.

The first one (cipher1.txt) is a Shift cipher with the shift key is 14.

The second one (cipher2.txt) is Substitution cipher which I have found the original plaintext. I’ll show the process of decrypting cipher2 manually below.

The third one (cipher3.txt) is a Vigenère cipher with the key turns out to be ‘tucson’.

The last one (cipher4.txt) is also a Vigenère cipher with key = ‘nebraska’.

For the Shift cipher, since we already got the key during the eliminating process so it only takes a little more effort to write the decrypt function for this cipher type.

For the Vigenère ciphers, the decrypting function needs more steps than the Shift cipher.

* For each , we have to break the ciphertext into substrings and compute the Index of Coincidence for each substring. Hence, I wrote another function to break a ciphertext and do the computations. After that, I would take the average of all the indices we just got. If there exists an such that the average is close to 0.06506, we take the first that has the average close to 0.06506 as the key length. For each that does not satisfy the condition, we just move to the next until we found one.
* Then after we got the key length, it means that we are gonna have shift keys.

To find the shift keys for each position for the key list, we go back and break the cipher ciphertext into substrings again.

For example:

ciphertext = “ABCDEFGHIJKLMN”

m = 3, subStr[1] = “ADGJM”, subStr[2] = “BEHKN”, subStr[3] = “CFIL”

Then we treat each substring as a Shift cipher, use the Shifted Index of Coincidence to find the key for each one. The shift key for the 1st substring is the first one of the key list, and we keep finding all shift keys to have a complete key list.

All the process above is already automated. My program would do all the works. It will print out the Frequency Analysis, together with the Index of Coincidence and the key to the terminal. But since the messages are too long, the decrypted ones will be outputted to .txt files, named by your request.

For the case of Permutation cipher, although we weren’t given any Permutation cipher to decrypt, but I was also trying to write a function to automate the decrypting process. However, with limited examples and time, I was only able to pull out the key length for some examples that I got. Then we would be able to generate a table by filling it column by column. I was not able to find the permutation of the columns hence haven’t completely decrypt any permutation ciphers.

Above is everything about my strategies for this project.

Keep reading to the instructions to run the program. The explanation for the decrypting ciphertext2 will also be found after the instructions.

**How to run the project:**

I use Python for this project, and the ciphertext should be inputted on as an argument when we compile the code in the terminal. Users can start the process by running the following command:

python main.py cipher.txt

where cipher.txt could be any file that contains your ciphertext. All the source code files and input files should be in a same folder.

After that, the program will analyze and decrypt the message. Once the process is complete, the message will be outputted to a text file. Users’ will be asked to name the text file from the terminal. Note that if you want to name it ‘plaintext.txt’, you should enter the whole thing ‘plaintext.txt’, not just ‘plaintext’.

Users can keep run the program again to analyze and decrypt other ciphers.

**What to find in the source code:**

In the .zip submission, multiple files will be found:

* main.py: this file is the main one, used to call functions from all other files to run.
* frequenceAnalysis.py: this file contains functions relating the frequency analysis (find the monogram frequencies and diagram frequencies)
* indexOfCoincidence.py: this file contains function (singleKeyIndex) to compute a simple IoC of a cipher (given the monogram frequencies which can be obtained by functions in frequenceAnalysis.py), function (indexOfCoincidence) to break the ciphertext into m substrings and compute the average IoC for these, function (shiftedIndexOfCoincidence) to compute the shifted IoC for a given cipher and a value of g.
* shiftedCipher.py: containing two functions, first is findShiftKey to find to shift key, second one is decryptedCaesar is to decrypt a given cipher from a given parameter key.
* vigenereCipher.py:
  + isVigenere is to determine the cipher type from the IoC.
  + findKeyLen is to find the key length for a Vigenere cipher, approach is already described.
  + multiKeysShift is to perform the shifting using multiple keys.
  + decryptedVigenere is to connect everything function above and return the complete decrypted message.
* permutationCipher.py:
  + isPermutation is to check if a given cipher is a Permutation cipher, by comparing the monogram frequencies in the cipher to English.
  + compareToEngDiFreq is to compare the most common diagrams frequencies in the cipher and in English, for decrypting purpose.
  + decryptedPermutation is to decrypt permutation ciphers but hasn’t reached the final message yet. But it could find the key length in some cases and return the corresponding matrix.

Besides, you can also find decryptingCipher2.txt in the .zip file containing all the keys and can generate the complete decrypted message and print to a txt file. Use the command to run:

python decryptingCipher2.txt cipher2.txt

Where cipher2.txt is the file containing the second substitution cipher.

In the .zip file, decrypted messages of cipher1, cipher2, cipher3, cipher4 can also be found in plaintext1.txt. plaintext2.txt, plaintext3.txt, plaintext4.txt respectively.

First, I did the Frequency Analysis and here are what I got:

Cipher VS English

T : 14.0 % e : 12.7 %

G : 10.3 % t : 9.1 %

Z : 7.6 % a : 8.2 %

Q : 7.5 % o : 7.5 %

D : 7.5 % i : 7.0 %

F : 7.3 % n : 6.8 %

R : 6.6 % s : 6.3 %

U : 5.8 % h : 6.1 %

P : 5.8 % r : 6.0 %

W : 4.2 % d : 4.3 %

Y : 3.4 % l : 4.0 %

A : 3.3 % u : 2.8 %

X : 2.8 % c : 2.8 %

H : 2.5 % w : 2.4 %

E : 2.0 % m : 2.4 %

C : 1.9 % f : 2.2 %

M : 1.7 % y : 2.0 %

O : 1.4 % g : 2.0 %

L : 1.2 % p : 1.9 %

J : 1.2 % b : 1.5 %

K : 1.0 % v : 1.0 %

N : 0.3 % k : 0.8 %

S : 0.2 % x : 0.2 %

B : 0.1 % j : 0.2 %

I : 0.1 % z : 0.1 %

V : 0.1 % q : 0.1 %

The Most Common Digrams In The Cipher Are:

GP : 676 times

PT : 573 times

TD : 524 times

UT : 357 times

QF : 344 times

GT : 340 times

QX : 340 times

TU : 331 times

ZG : 328 times

TF : 320 times

ZF : 311 times

RF : 296 times

ZW : 296 times

GR : 288 times

FG : 281 times

Then from <https://cryptoclub.org/#vAllTools>, I found out that most common trigrams are:

GPT, ZWW, DPZ, PZW, ZFY.

From this information, I immediately matched GPT to ‘the’, ZFY to ‘and’. ZWW got me thinking for a few second before I tried ‘all’ with ZWW.

At this point, the first few characters in the cipher turn out to be:

“KetheCeQCleQXtheHnRtedDtateD…” (1)

Next we have Q and D are the third and fourth common letters while TD, QF are the third and the fifth common digrams in the cipher.

Then from this page, I found list of the most common digrams in English, which is

th, he, in, er, an, re, on, at, en, nd, ti, es, or, etc.

Then D could be ‘r’ or ‘s’. And from the partial message we have, I decided to match D to ‘s’ to have to word ‘states’ which leads me to the famous words “We, The People Of The United States”

“wethepeopleoftheunitedstates…”

From now on, it gets easier. A bit after the previous phrase that we just decoded, I found

“…unionestaMlish…”

which leads M to ‘b’.

Later in the paragraph, we have an almost completed phrase of “the constitution for the United States of American” which helped me solve for ‘c’, ‘r’, ‘m’.

From here, we already have enough information, we could read the rest to fill out to the keys.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| Z | M | A | Y | T | X | L | P | R | S | B | W | E | F | Q | C | I | U | D | G | H | J | K | N | O | V |

I quickly write a simple program to decrypt the message using this key table.

Here are a partial decrypted message. I’ll attach the full text stored the file “plaintext2.txt” in my submission.

“wethepeopleoftheunitedstatesinordertoformamoreperfectunionestablishjusticeinsuredomestictranquilityprovideforthecommondefencepromotethegeneralwelfareandsecuretheblessingsoflibertytoourselvesandourposteritydoordainandestablishthisconstitutionfortheunitedstatesofamericaarticleisectionalllegislativepowershereingrantedshallbevestedinacongressoftheunitedstateswhichshallconsistofasenateandhouseofrepresentativessectionthehouseofrepresentativesshallbecomposedofmemberschoseneverysecondyearbythepeopleoftheseveralstatesandtheelectorsineachstateshallhavethequalificationsrequisiteforelectorsofthemostnumerousbranchofthestatelegislaturenopersonshallbearepresentativewhoshallnothaveattainedtotheageoftwentyfiveyearsandbeensevenyearsacitizenoftheunitedstatesandwhoshallnotwhenelectedbeaninhabitantofthatstateinwhichheshallbechosenrepresentativesanddirecttaxesshallbeapportionedamongtheseveralstateswhichmaybeincludedwithinthisunionaccordingtotheirrespectivenumberswhichshallbedeterminedbyaddingtothewholenumberoffreepersonsincludingthoseboundtoserviceforatermofyearsandexcludingindiansnottaxedthreefifthsofallotherpersons…”