CS411 Cryptography Homework #1
Hakan Buğra Erentuğ 23637
Erkay Savaş 12 / 10 / 19

Before starting:

The all python codes that are attached, are my own work, unless I stated otherwise. You can run and check my results in your own computer to verify the solutions. I did not focus on comments because the codes that I designed is just helping me to find solutions. Because any code grading is not the case as far as I know. However, If you see anything suspicious, inform me and I can assure you I could explain what I did. You can find my python codes at the zip file. I also publish my codes on GitHub, to guarantee myself in case of any plagiarism. https://github.com/herentug/CS411

Solution #1

To encode cipher text, we can brute attack because the key space is very small. The trying of 26 different keys manually, will bring solution.

The outputs of 'shiftcipher.py' are placed below.

['nzwo', 'oaxp', 'pbyq', 'qczr', 'rdas', 'sebt', 'tfcu', 'ugdv', 'vhew', 'wifx', 'xjgy', 'ykhz', 'zlia', 'amjb', 'bnkc', '**cold**', 'dpme', 'eqnf', '**frog**', 'gsph', 'htqi', 'iurj', 'jvsk', 'kwtl', 'lxum', 'myvn']

Apparently, frog and cold are two meaningful English word which obtained from decryption.

Solution #2

What we should check first is frequency analysis of letters. To achieve that, I write a code which name freqanalisis.py. The output is the following...

```
A: 0.0208333333333
C: 0.0625
B: 0.0416666666667
E: 0.0104166666667
D: 0.0104166666667
G: 0.0
F: 0.0625
I: 0.0416666666667
H: 0.0833333333333
K: 0.0
J: 0.0625
M: 0.0729166666667
L: 0.0
O: 0.0
N: 0.0104166666667
Q: 0.0
P: 0.0
S: 0.03125
R: 0.0625
U: 0.0625
T: 0.03125
W: 0.0104166666667
V: 0.0520833333333
Y: 0.0104166666667
X: 0.03125
Z: 0.03125
```

CS411 Cryptography Homework #1
Hakan Buğra Erentuğ 23637
Erkay Savaş 12 / 10 / 19

A : ||
C : ||||||
B : ||||
E : |
D : |
G :
F : ||||||
I : ||||
H : ||||||||
K :
J : ||||||
M : |||||||
L :
O :
N : |
Q :
P :
S : |||
R : ||||||
U : ||||||
T : |||
W : |
V : |||||
Y : |
X : |||
Z : |||

The hint that the most frequent letter in the plaintext is “A”, has already given. Therefore we can assume C,F,H,J,M,R,U,V which are most frequent letters could be decrypted to letter ‘A’. In addition, the person who crypt it, made another mistake and forgot the delete spaces between words. In english, I and A is the one letter words, therefore we can assume ‘A’ letter will be encrypted to ‘H’. Luckily, In affine cipher, key space is also very low. Therefore we may check approximately 300 keys and filter them by looking first letter of the text. The outputs of code ‘affinecipher.py’, has given below.

```
11-1    :a eqmmyeerqb uan we gny cdg man bao a rwlu rgqnfaxwgn cwxd xdy tlwmie gxdyle dajy xdlgcn ax dwu.
7-3     :a mwkkummzwd ian om snu gjs kan daq a zohi zswnparosn gorj rju fhokym srjuhm jabu rjhsgn ar joi.
3-5     :a uciiquuhcf wan gu enq kpe ian fas a hgdw hecnzalgen kglp lpq rdgiou elpqdu patq lpdekn al pgw.
25-7    :a ciggmccpih kan yc qnm ovq gan hau a pyzk pqinjafyqn oyfv fvm dzygec qfvmzc valm fvzqon af vyk.
17-11   :a successful man is one who can lay a firm foundation with the bricks others have thrown at him.
5-17    :a qmwwsqqdmr can kq yns izy wan rae a dkfc dymnhabkyn ikbz bzs lfkwgq ybzsfq zaxs bzfyin ab zkc.
23-21   :a gysskggtyv ean ug wnk qlw san vai a tuxe twynbapuwn qupl plk jxusmg wplkxg lahk plxwqn ap lue.
19-23   :a oeqqgoobex san mo ing uri qan xak a bmts bienlajmin umjr jrg vtmqco ijrgto razg jrtiun aj rms.
```

The run time is 0.86 ms. We obtained 8 output and we can easily detect the plain text with **17-11** gamma-theta values: **a successful man is one who can lay a firm foundation with the bricks others have thrown at him.** However, these are decryption key and we hasn't found encryption key yet. Then I write overwritten similar code which is with new alpha and beta values. Since we know the plain and cipher text it is easy process. The correct alpha and beta values are **23 – 7.**

Solution #2 without Exhaustive Search

After we found the letter a which has index 0. Due to formula "a*x+b = y" and x=0, a becomes irrelevant. Therefore b=y or in other saying 'h' index 7 could be concluded. After the solution of beta, alpha could take less than 13 value which can be easily calculated. Other hints like, 'hc' bi-gram word may correspond to easy-to-find words such as 'at', 'an' or 'am'. After the solution of alpha and beta values, rest of the solution can be calculated easily.

Solution #3

What we are trying to do is, combining each letter and create an new alphabet of bigrams. There is 31 letter in the alphabet that are told. Therefore total permutation of these letters are 961 (31*31). Please check bigramDictionary.ipynb for python notebook code that I used to create them. The output is stored in bigram.txt text file. Therefore we have to modulo 961 to not express the limit. Key space is little bit harder, obviously, beta value could take anything between 0 to 961, because it just shift the alphabet. However, alpha could not take any value that has gcd(a,b) != 1. Luckily, 31 is a prime number therefore 961 has not factor expect 31. That means a only could not take the factors of 31. It cannot take a=31k where k is a integer between 1 to 31. That means, 961-31= 930 for alpha value. The total key space is 961*930=893730

Solution #4

I already implement affine cipher on solution 2. However, I must change some values to adjust the new alphabet. Please check bigram.py for my implementation, and bigram.txt for my bigram dictionary. Exhaustively, we can search 893730 key pair, however, In the question 3, there was a hint which obviously given. If the plain text has odd numbered length, we put X at the end of it. Therefore we have 50% chance to 'YT' the last element of cipher-text, is going to decrypt to the '*X'. Therefore I added some filters to my output calculator. I also eliminate question marks and exclamations before last 2 element, because they can only be in the end of the file. (point and comma could be in the middle of sentence). Then I manually, trying to see something similar to my eyes between 48000 outputs which you can find in output.txt file.

```
651-23     :TXLXIXTXIXYXHXGXIXJXUXCXOXJXUXRXPXXXHXYXVXIXKXFX.X
287-26     :QRALMBURPBGNBIOPMBZZAKTMNVZZMKZANOSUHIINNDSBWSQW.X
161-64     :HDVOWWKDBWMAMETRWWMJUVNHAGMJ,VX,ZY NBE,ARILW..K?.X
831-106    :YMRBGYVM.YDPOLD,GYUGNZ,IQJUGEZBRYWRCZLRPUHRYHTJQ.X
341-147    :LX XUXLXUXVXSXQXUXWXNXIXBXWXNXHXDXTXSXVXPXUXYXOX.X
79-165     :XSUNJPQS.PEZK BGJP EPHPT L EZHUJVAUFP QZZ.OPCYSR.X
477-167    :VOXFLV.OAVNIGQGLLVF BT WZUF TTZE,Z.DPQQIJYUVVAFG.X
730-289    :PSENVPISIPBZV LGVPIEIHVTNLIESHKJJAQF  NZT. PQY.R.X
167-291    :NOHFXVTOMVKIRQQLXVS ZTBWMUS MTPERZXD QNIDYBVEAOG.X
201-296    :IE,Q,FAEXFAMGW,I,FOTWS,OK.OT,S.HYKY,ZW.MKBRFDCLZ.X
934-319    :OUORXMQUJMESTA TXMNYVBCCCWNY.BU.RD.GWAFS N MLFNH.X
473-336    :.RWLLBAROBONNIIPLBLZ,KDM.VLZKKAAOOIUTIQN,DRB SXW.X
672-337    : PIHMEVPQESULDLCMERFWQI,.KRFHQCNDL.TTDAUFRUEUGRB.X
```

CS411 Cryptography Homework #1
Hakan Buğra Erentuğ 23637
Erkay Savaş 12 / 10 / 19

```
876-367     :ZRSLOB,RRBVNII POBWZEKUMQVWZQKNALOHUOIXNMDUBOSCW.X
779-381     :KUGR,MMUPMSSJAAT,MEYCBFCLWEYIBP.LDZGMATSXNBMSFCH.X
449-389     :CERQHFZEBFVMWWVIHFQTJSSOI.QTQSEHPKV,KWRMVB F,CKZ.X
823-444     :FSPNFP,SXPFZ. IGFPBEHHNTULBERHNJZALFB RZ..KPSYPR.X
697-482     :.EFQQFTEKFLMHWNIQFST.SHOG.STDSMHGKS, WHMBBEFYCJZ.X
37-498      :KD.OCWNDMWRAEEXRCWLJLVDHBGLJUVT,OYMNYEDA.IWWO. ?.X
139-513     :ZEBQTFRENFSMCWAITF,TCSYO .,TJSZHDKR,VWOMPBHFMCTZ.X
411-553     :NRZLBBRREBBNJIKPBB ZJK,MMV ZVK,AYOBUPIDNDDHBESAW.X
916-599     : S NUPTSHPJZC FGUPZEGHFT.LZEQHQJKAGFH VZE.ZPUYDR.X
62-643      :KXYXGXKXGXJXAXZXGXMXQXBXLXMXQX,XRXDXAXJXWXGXSXTX.X
465-674     :IXUXJXIXJXQX XMXJXXXWXSXAXXXWXLXOXCX XQXFXJXGX,X.X
635-699     :NEIQGFFEAF,MDWPIGFCTHSCOW.CTOSKHQKL,WWZMGBZFCCRZ.X
407-722     :TUYRAMVURMCSQAMTAMBYGBGCOWBYMBD.JDOGTADSWNDMKFTH.X
480-761     :JEAQMFBEGFMMYWUIMFYTTSFOA.YT SFHKKJ,MWIMDBAFJCGZ.X
451-785     :OSCNHPHSZPUZD UGHP,ELHOTXL,EVHBJXAAFI BZ .MPKYBR.X
393-833     :ZPGH,EUPCEGUYDUC,EHFZQB,GKHFKQYNRLLTBDTUMRGEOGWB.X
626-839     :TDOOEWWDOWBALEEREWIJPVEHEGIJYVH,MYBNAESA IYWG.M?.X
296-847     :KSZNNPDSAPDZY ZGNPUEXHRTBLUECH.JRA,F, PZX.SPRYVR.X
238-895     :VP,HEEQPIEUUODZCEE,FGQE,PK,FWQTNLLJTWDCUJRMEVGLB.X
903-908     :HUARSMJUEMNSRA.TSMFYLBPCKWFYRBT.WDIGUAOSNNVMAFRH.X
180-943     :BMCBZY,MPYGPDLY,ZYHGUZXI,JHGLZLRFWVCOLUP HFYYTAQ.X
248-953     :VXPXFXVXFXRXMXTXFX,XOXQXZX,XOXEXSXYXMXRXHXFXWX X.X
```

Above, you can see some of the examples of the output. And then I started to try some quadrams which are common in english according to "http://practicalcryptography.com/cryptanalysis/letter-frequencies-various-languages/english-letter-frequencies/", and then nothing is concluded. There are two probability that's why I cannot decode. Firstly, my dictionary files are not correct. Secondly, I actually found the plain text but cannot detect between these outputs. After I learn the plain text, I am going to check. Still, I am going to attach output.txt file without any filtering mechanism. Update: The file was so big to upload, so I take only 48000 row of data.

“”

Update 2: I keep researching and found my mistake. As I assumed, I miss-indexing the dictionary of bigrams in bigramDictiony.ipynb. The following correct bigram dictionary code is taken from internet:

```
alp = ['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', ' ', '.',\
         ',', '!', '?']

m = {}
inv_m = {}
count = 0
for i in range(len(alp)):
    for j in range(len(alp)):
        m[single_letter[i]+alp[j]] = count
        inv_m[count] = alp[i]+alp[j]
        count += 1
```

and the plaintext is **THOSE WHO BELIEVE IN TELEKINETICS, RAISE MY HAND..**

I did cleary state the parts after update 2 is not my own work to prevent plagiarism. I accept my mistake, and reduction of my points.

CS411 Cryptography Homework #1
Hakan Buğra Erentuğ 23637
Erkay Savaş 12 / 10 / 19

Solution #5

In that question, we did a easy implementation of vinegere cipher. I attached the code that I used that named vingerecipher.py, in order to decrypt the cipher text "A RRNNQW TB IGQOEE BAYL QHMLRAOA WG RZE TZHSFDF BAYL I QWG'R CNBE MFW AAAPCJ." What I do is subtracting the key word indices from the cipher-text in-orderly. Please see vingerecipher.py for the python implementation. The output is “**I REFUSE TO ANSWER THAT QUESTION ON THE GROUNDS THAT I DON'T KNOW THE ANSWER**”
.

Solution #6

The first task is finding the key length. To achieve that we have to shift the text by one and compare with itself and count coincides. For python code that I write to do this please check “keylength.py”. After I count coincides, the output was:
[0, 20, 20, 16, 22, 16, 15, 27, 12, 16, 15, 19, 19, 20, 34, 21, 20, 13, 6, 13, 24, 31, 25, 17, 12, 20]. Therefore max 3 value are 34,31 and 27. However, 34’s index is 14 and 31’s index is 21, or other saying key lengths which are too long to remember. The index of 27 which is third largest coincides and factor of other 2 is 7. Therefore, it would be wise to think key length is 7. (In fact, I tried 14, create subtexts and apply letter frequency analisis but could not anything. So I passed to 7). Please check the vigeretoshift.py to learn how to parser the plain text to subtextes. These are the outputs of 14 key length subtexts that I found in my first attempt:

```
['gogrroypwebklcdsqckwvsvzkdlldrd', 'slspsdhtjdeqplsytmdpjrsuyosssdo',
'orivetalltoatnegvekmopypgadwdza', 'oaoeskpheponszzkaheepsdtebeelg',
'mqptqxkcbpbumninatnapswdhqooak', 'olgddhzhlagzwjjlzkllwmvmgbbbmj',
'ngtzkhgvkmonmnjghzgzqwvovagghw', 'yscqxxssougedjooccoucocnzgjfze',
'oyppzrdesttdspdjsscpzyyybkoyxy', 'skitthwhotteenteaaetuqhovgpmoc',
'peodoakonmdjshawngedjensaepeme', 'psqwmzwvacwmwidzvmamlwuwjlwbee',
"wffmwwvgwwmsgcwzwlkg'anzlzvwta", 'rnsflsrvldsqcdmdrnnmkqdwvgvjwd']
```

And the correct subtexts are with length 7.

```
['gyosgcrqrxoxyspswoeubgkeldcjdosoqccckowuvcsovcznkzdgljlfdzred',
'solyspppszdrhdtejsdtetqdpslpsdyjtsmsdcppjzrysyuyyboksosysxdyo',
'osrkiivtetthawlhlottotaetennetgevaeakemtoupqyhpogvagdpwmdozca',
'opaeooedsokapkhoenpmodnjsszhzakwanhgeeedpjsedntseabeepeelmge',
'mpqspqtwqmxzkwcvbapcbwummwniidnzavtmnaamplswwudwhjqlowobaeke',
"owlfgfdmdwhwzvhglwawgmzswgjcjwlzzwkllklgw'mavnmzglbzbvbwmtja",
'nrgntszfklhsgrvvklmdosnqmcndjmgdhrzngnzmqkwqvdowvvaggvgjhwwd']
```

you can check and verify by vigeretoshift.py.

Then, we have to letter frequency analysis for each subtext. Unfortunately, I had overwritten on my code again and again to calculate seven different subtext. Therefore, I will take one sample to explain the process. Below, I did letter frequency analisis with freqanalisis.py for first subtext “gyosgcrqrxoxyspswoeubgkeldcjdosoqccckowuvcsovcznkzdgljlfdzred”. Apparently, C,D,G,O and S frequently repeated. Therefore it is wise to think these corresponds to most frequent letters, E,A,N,O,T,L. Then, to check best option I write an algorithm to calculate fitness level between our letter frequency and English alphabet’s. Please check vigeretoshift.py The fitness level has calculated from mean square error and best options for k[1] are Q,T and U. Of course, we tried Q first.

```
A       :
C       : |||||||
B       : |
E       : |||
D       : |||||
G       : ||||
F       : |
I      :
H       :
K       : |||
J      : ||
M       :
L       : |||
O       : |||||||
N       : |
Q       : ||
P       : |
S       : |||||
R       : |||
U       : ||
T       :
W       : ||
V       : ||
Y       : ||
X       : ||
Z       : |||
```

After I did the previous step for the each subtext, I found the sequence k1=q, k2=p, k3=a, k4=e, k5=s, k6=i and k7= b. Key is k={qpaesib}. Below is the encoded, sequence of plain text. The plain text is the following.

“whosewoodstheseareithinkiknowhishouseisinthevillagethoughhewillnotseemes toppingheretowatchhiswoodsfillupwithsnowmylittlehorsemustthinkitqueertosto pwithoutafarmhousenearbetweenthewoodsandfrozenlakethedarkesteveningofth eyearhegiveshisharnessbellsashaketoaskifthereissomemistaketheonlyothersoun d'sthesweepofeasywindanddownyflakethewoodsarelovelydarkanddeepbutihavep romisestokeepandmilestogobeforeisleepandmilestogobeforeisleep”

I googled it:

Whose woods these are I think I know.
His house is in the village though;
He will not see me stopping here
To watch his woods fill up with snow.

My little horse must think it queer
To stop without a farmhouse near
Between the woods and frozen lake

CS411 Cryptography Homework #1
Hakan Buğra Erentuğ 23637
Erkay Savaş 12 / 10 / 19

The darkest evening of the year.

He gives his harness bells a shake
To ask if there is some mistake.
The only other sound’s the sweep
Of easy wind and downy flake.

The woods are lovely, dark and deep,
But I have promises to keep,
And miles to go before I sleep,
And miles to go before I sleep.