CSCI 325 HW 2

Wesley Smith

February 9, 2024

1 With key

```
>>> exec(open("cipher.py").read())
>>> ciphertext = "omitted"
>>> devc(ciphertext, "TAGORE")
straybirdsofsummercometomywindowtosingandflyawayandyellowleavesofautumnwhi
```

The code for this and the next part is submitted with this file as cipher.py. I recommend running inside the python interpreter so you can directly call the functions as just assign the cipher text and useful items as variables in there.

2 Cipher text only

1. First we need to figure out the key length, to do that I made a function that printed out the IoC at varying periods.

```
>>> ioc_rainbow(ciphertext, 18)
IoCs if key len is 1: 0.041,
IoCs if key len is 2: 0.0412, 0.0408,
IoCs if key len is 3: 0.0465, 0.0505, 0.0503,
IoCs if key len is 4: 0.0399, 0.0408, 0.0425, 0.0415,
IoCs if key len is 5: 0.043, 0.0392, 0.0392, 0.0438, 0.041,
IoCs if key len is 6: 0.0469, 0.045, 0.0501, 0.0447, 0.0545, 0.0497,
IoCs if key len is 7: 0.0363, 0.043, 0.0398, 0.0407, 0.0416, 0.0395,
    0.04,
IoCs if key len is 8: 0.0431, 0.0404, 0.0455, 0.0437, 0.0374, 0.0431,
     0.041, 0.0437,
IoCs if key len is 9: 0.0684, 0.0735, 0.0765, 0.066, 0.0686, 0.0601,
    0.0632, 0.0627, 0.0693,
IoCs if key len is 10: 0.0432, 0.041, 0.0394, 0.0442, 0.043, 0.0396,
    0.0362, 0.0356, 0.0469, 0.0402,
IoCs if key len is 11: 0.0437, 0.0408, 0.0403, 0.0408, 0.0359, 0.041,
     0.0408, 0.042, 0.0481, 0.0388, 0.0409,
IoCs if key len is 12: 0.0453, 0.0396, 0.0525, 0.0442, 0.0564,
    0.0447, 0.0441, 0.0491, 0.0497, 0.045, 0.052, 0.0544,
IoCs if key len is 13: 0.0407, 0.0393, 0.0431, 0.0441, 0.0369,
    0.0373, 0.0403, 0.0434, 0.0427, 0.04, 0.0431, 0.0403, 0.0386,
IoCs if key len is 14: 0.0301, 0.0446, 0.0411, 0.0415, 0.0415, 0.036,
     0.0386, 0.0366, 0.0423, 0.0398, 0.0431, 0.0406, 0.0447, 0.0366,
IoCs if key len is 15: 0.0484, 0.0452, 0.0461, 0.0484, 0.0452,
    0.0488, 0.0457, 0.0443, 0.0516, 0.0611, 0.0448, 0.0485, 0.0466,
    0.0559, 0.0569,
IoCs if key len is 16: 0.0471, 0.0425, 0.041, 0.042, 0.0369, 0.044,
    0.0353, 0.0548, 0.037, 0.0391, 0.0613, 0.0449, 0.0344, 0.0534,
    0.0502, 0.0391,
```

```
IoCs if key len is 17: 0.0339, 0.0386, 0.0415, 0.0427, 0.0456,
    0.0415, 0.0462, 0.0386, 0.0327, 0.0339, 0.0526, 0.0415, 0.0345,
    0.0514, 0.0357, 0.0436, 0.052,
IoCs if key len is 18: 0.0597, 0.0688, 0.076, 0.0591, 0.0818, 0.0519,
     0.0799, 0.0506, 0.0643, 0.0747, 0.0687, 0.0727, 0.066, 0.0646,
    0.0626, 0.0505, 0.0667, 0.0694,
```

As seen from the prior output the IoC is very close to English (0.067) at 9 and 18 and 18 being just 2x 9 the key length is almost guaranteed to be 9.

2. Next my focus was to guess the key. To do that I turned to letter frequencies. I wrote a function that gave a most probable key, it took the letter frequencies for each bucket where the Cesar cipher would be the same and gave the shift to make the most common letter E. To use do optimal_key(bucket(ciphertext, 9)). This resulted in the key: MPRKTGAXR. Sidenote, after perfroming the steps listed bellow and determining the key I wrote a slightly smarter function for getting they key which looks at more than just e to guess it. Running more_optimal_key(bucket(ciphertext, 9)) results in MARKTWAIR.

3. Decryption based on that key did not give a coherent text however it did seem to have some more patterns in it than before to I figured I'd experiment with it. First I realised if that the key was an english word it would likely have more vowels and P is a very odd letter for the second letter of the word. I decided to look at the letter frequencies for the second bucket. To do that I used the function I wrote.

```
>>> get_let_freq(bucket(ciphertext, 9)[1])
A, occurs 10 with a freq of 0.09009009009009009 : 22 (W) -> E
B, occurs 1 with a freq of 0.009009009009009009 : 23 (X) -> E
C, occurs 4 with a freq of 0.036036036036036036 : 24 (Y) -> E
D, occurs 1 with a freq of 0.009009009009009009 : 25 (Z) -> E
E, occurs 14 with a freq of 0.12612612612612611 : 0 (A) -> E
F, occurs 3 with a freq of 0.02702702702702703 : 1 (B) -> E
G, occurs 2 with a freq of 0.018018018018018018 : 2 (C) -> E
H, occurs 12 with a freq of 0.10810810810810811 : 3 (D) -> E
I, occurs 5 with a freq of 0.04504504504504504 : 4 (E) -> E
J, occurs 0 with a freq of 0.0 : 5 (F) -> E
K, occurs 1 with a freq of 0.009009009009009009 : 6 (G) -> E
L, occurs 8 with a freq of 0.07207207207207207 : 7 (H) -> E
M, occurs 2 with a freq of 0.018018018018018018 : 8 (I) -> E
N, occurs 2 with a freq of 0.018018018018018018 : 9 (J) -> E
O, occurs 7 with a freq of 0.06306306306306306 : 10 (K) -> E
P, occurs 2 with a freq of 0.018018018018018018 : 11 (L) -> E
Q, occurs 1 with a freq of 0.009009009009009009 : 12 (M) -> E
R, occurs 10 with a freq of 0.09009009009009009 : 13 (N) -> E
S, occurs 4 with a freq of 0.036036036036036036 : 14 (O) -> E
T, occurs 16 with a freq of 0.14414414414414414 : 15 (P) -> E
U, occurs 1 with a freq of 0.009009009009009009 : 16 (Q) -> E
V, occurs 0 with a freq of 0.0 : 17 (R) -> E
W, occurs 3 with a freq of 0.02702702702702703 : 18 (S) -> E
X, occurs 0 with a freq of 0.0 : 19 (T) -> E
Y, occurs 2 with a freq of 0.018018018018018018 : 20 (U) -> E
Z, occurs 0 with a freq of 0.0 : 21 (V) -> E
```

Looking at the prior frequencies I noticed that E occurs a close second to T and the I would have to shift 0 spots using an A, a vowel(!) so I decided to decode the cipher text with the P an A. This left be with this partially decrypted text:

```
>>> cdevc(ct, "MARKTGAXR")
thegobdpjeraseftpiberityslcoodthynrlerhapitzsriteferedeamusumpjtofthupfxlicbuj
ieesafarxirderanddomherthidgekwritevocpheiridsenuctiodtsairproviephei ....
```

I made a modified version of my decoding function cdevc which underlines the period of the key. Unfortunately the styling does not show here but using that I noticed the first 5 letters of the key often made coherent words or English syllables like "**write**vocp", "**house**madx", "d**then**vow", etc

4. I then looked at the letter frequencies for the 6th bucket of letters and tried a couple of them and W seemed to make the most sense based on freq and looking at patterns in the words ie "uctio**n**tsa", I knew that probably had to N as -tion and W gave that.

5. at this point with my key as "MARKTWAXR" I just took a guess that it was Mark Twain and it worked!

6. The decrypted text follows:

```
cdevc(ct, "MARKTWAIN")
thegoldeneraseptemberitisagoodthingperhapstowritefortheamusementof
thepublicbutitisafarhigherandnoblerthingtowritefortheirinstructio
ntheirprofittheiractualandtangiblebenefitthelatteristhesoleobject
ofthisarticleifitprovethemeansofrestoringtohealthonesolitarysuffe
reramongmyraceoflightinguponcemorethefireofhopeandjoyinhisfadedey
esofbringingbacktohisdeadheartagainthequickgenerousimpulsesofothe
rdaysishallbeamplyrewardedformylabormysoulwillbepermeatedwiththes
acreddelightachristianfeelswhenhehasdoneagoodunselfishdeedhavingl
edapureandblamelesslifeiamjustifiedinbelievingthatnomanwhoknowsme
willrejectthesuggestionsiamabouttomakeoutoffearthatiamtryingtodec
eivehimletthepublicdoitselfthehonortoreadmyexperienceindoctoringa
coldashereinsetforthandthenfollowinmyfootstepswhenthewhitehousewa
sburnedinvirginiailostmyhomemyhappinessmyconstitutionandmytrunkth
elossofthetwofirstnamedarticleswasamatterofnogreatconsequencesinc
eahomewithoutamotherorasisteroradistantyoungfemalerelativeinittor
emindyoubyputtingyoursoi
```