COL 759

Assignment-1 Part-2 Report

Pushpam Anand (2016CS10347)

Assignment - Given the ciphertext and length of the key, perform cryptanalysis to find the plain text and the key. Test on different cases.

Introduction -

- Cryptanalysis of hill cipher is done by assuming that key size is n=2 or n=3.
- So, most common bi-grams or tri-grams are used accordingly.
- " *Sympy* " library in python is used for the function of modulo-inverse.
- Cryptanalysis for both key-size 2 and 3 is done.

Cryptanalysis Algorithm Used:

1. Let's say key **K** has size, n = 2. Assume a pair of most common di-grams in english.
2. Make the matrix **P**, using the most common di-gram pair in english ("th" and "of ").
3. Find the most common di-gram pair in cipher text. Make a matrix **C**
4. Use the formula, $\mathbf{K = C \times P^{-1}}$ to find the key, **K**
5. Find the decrypted plaintext using the key, and the decryption algorithm done in assignment_1
6. If the decryption is not correct then change the di-gram pair in step-2 and repeat.
7. If analysing using many different di-grams in a loop, use IC of the plaintext as a stopping criterion.

Explaining the Code :

The code is done in python using jupyter notebook, and numpy and sympy libraries are used. Here functions are explained one-by one.

- This function makes the matrix, **P** using the most common digraphs.
  Usage: make_two_key("th", "he")

```
def make_two_key(first_most,second_most):
    key = np.zeros(4).reshape(2,2)
    key[0][0] = ord(first_most[0]) - ord('a')
    key[0][1] = ord(first_most[1]) - ord('a')
    key[1][0] = ord(second_most[0]) - ord('a')
    key[1][1] = ord(second_most[1]) - ord('a')
    return key.astype(int)
```

-

```
def most_common_digrams(cipher):
    mdict = {}
    input_text = list(cipher)
    input_text = list(filter((' ').__ne__, input_text))
    for i in range(0,len(input_text)-1,2):
        s = input_text[i] + input_text[i+1]
        if s not in mdict:
            mdict[s] = 1
        else:
            mdict[s] +=1
    sorted_x = sorted(mdict.items(), key=lambda kv: kv[1],reverse = True)
    a = sorted_x[0][0]
    b = sorted_x[1][0]
    print(a)
    print(b)
    return make_two_key(a,b)
```

The above function takes the cipher_text as input and finds the most common di-grams in the cipher text and returns the matrix **C** using them.

Usage :

```
In [38]: most_common_digrams(cipher_text)

         most common digrams in the cipher text are
         el
         th

Out[38]: array([[ 4, 11],
                [19,  7]])
```

- Code to compute the IC of any given string.

```
def IC(text):
    mdict = {}
    l = 0
    for a in text:
        l+=1
        if a not in mdict:
            mdict[a] = 1
        else:
            mdict[a] +=1
    s = 0
    for a in mdict:
        s += mdict[a]*(mdict[a]-1)
    return (1.0*s)/(1.0*l*(l-1))
```

- **The following code is well- commented for better understanding.**

  The following is the **decrypt function** for the key of size =2. In this, the cipher text is input as an argument and it returns the plain text. First the most common di-gram are assumed and matrix P is made, then matrix C is made from the most common di-gram in cipher text. Then key,K is computed and then the cipher is decrypted to give plaintext. IC of the plaintext is computed.

```
# decrypt function for keysize = 2, argument "raw_cipher" is the cipher text.
def decrypt_2(raw_cipher,n=2):
    a = "el"  # assumed most common di-gram
    b = "th"  # assumed most common di-gram
    C = most_common_digrams(raw_cipher).T    # make the matrix C
    P = make_two_key(a,b).T                  # make the matrix P from a,b

    # prepare_text func. makes set of two from the cipher text
    # for decrypting using the obtained key.
    cipher = prepare_text(raw_cipher,2)

    #following code computes the key, K using P and C
    temp = Matrix(P)
    Pinv = temp.inv_mod(26)    # P_inverse
    key = np.mod(np.dot(C,Pinv),26).astype(int)  #finding the key, using K = C x P_inverse
    print("encryption key is : ")
    print(key)
    tempkey = Matrix(key)
    kinv = tempkey.inv_mod(26)

    # finding the decrypted text using the obtained key, K_inverse
    # for every block of the text of size 2
    dec = []
    dec_text = ""
    for a in range(len(cipher)):
        b = np.dot(kinv,cipher[a])
        for i in range(len(b)):
            dec += [b[i]%26]

    # the following block of code is to put spaces at corresponding places.
    ind = 0
    for i in range(len(raw_cipher)):
        if(raw_cipher[i]!=' '):
            dec_text += chr(ord('a') + int(dec[ind]))
            ind +=1
        else:
            dec_text += ' '

    print("IC of the decrypted text is : ",IC(dec_text))
    return dec_text
```

How to do cryptanalysis using the Code :

Example :

- **Plaintext :** the electron is a subatomic particle symbol e or whose electric charge is negative one elementary charge electrons belong to the first generation of the lepton particle family and are generally thought to be elementary particles because they have no known components or substructure the electron has a mass that is approximately that of the proton quantum mechanical properties of the electron include an intrinsic angular momentum spin of a halfinteger value expressed in units of the reduced planck constant being fermions no two electrons can occupy the same quantum state in accordance with the pauli exclusion principle like all elementary particles electrons exhibit properties of both particles and waves they can collide with other particles and can be diffracted like light the wave properties of electrons are easier to observe with experiments than those of other particles like neutrons and protons because electrons have a lower mass and hence a longer de broglie wavelength for a given energy electrons play an essential role in numerous physical phenomena such as electricity magnetism chemistry and thermal conductivity and they also participate in gravitational electromagnetic and weak interactions since an electron has charge it has a surrounding electric field and if that electron is moving relative to an observer said observer will observe it to generate a magnetic field electromagnetic fields produced from other sources will affect the motion of an electron according to the lorentz force law electrons radiate or absorb energy in the form of photons when they are accelerated laboratory instruments are capable of trapping individual electrons as well as electron plasma by the use of electromagnetic fields special telescopes can detect electron plasma in outer space electrons are involved in many applications such as electronics welding cathode ray tubes electron microscopes radiation therapy lasers gaseous ionization detectors and particle accelerators interactions involving electrons with other subatomic particles are of interest in fields such as chemistry and nuclear physics the coulomb force interaction between the positive protons within atomic nuclei and the negative electrons without allows the composition of the two known as atoms ionization or differences in the proportions of negative electrons versus positive nuclei changes the binding energy of an atomic system the exchange or sharing of the electrons between two or more atoms is the main cause of chemical bonding in british natural philosopher richard laming first hypothesized the concept of an indivisible quantity of electric charge to explain the chemical properties of atoms irish physicist george johnstone stoney named this charge electron in and j j thomson and his team of british physicists identified it as a particle in electrons can also participate in nuclear reactions such as nucleosynthesis in stars where they are known as beta particles electrons can be created through beta decay of radioactive isotopes and in highenergy collisions for instance when cosmic rays enter the atmosphere the antiparticle of the electron is called the positron it is identical to the electron except that it carries electrical and other charges of the opposite sign when an electron collides with a positron both particles can be annihilated producing gamma ray photons

- This plaintext is encrypted using the key, K = 22 3 9 6

- **Cipher Text:** xfw iutxcady jg g owwjsvcal ezyalzgm oscmpu t vu lgyaw iutxcity eylchi gt gcufkfqa qml ryuexkcpel nizyoa ryqwbnjwj mryjwh mb gkj epmbu zxkjifkkaq tl dkj utxpjw sfphagut gtcacj nao bwv oamlkxpjn sorqidv sv ih ryuexkcpel sfphagutj mqwiqse xfey ylgf qt zmkyg zgqilmlfxw mm bbelqsnxcxwp ukj ryqwbnjw ylg g eeik xffk wy tmrdneeofkryn syls vl dkj rdbgjw wenakfo yqwylyjssb hadedphgsw ml dkj ryqwbnjw huzghqk ky jfxitcrag naksivu rgqxkkfg iqbq tg t ylxzhuonoav thokw bsrdmosem xi dyjet la xfj itcektc ztnawa iycrcpfx ihhur gjicajwt gb ggw ryqwbnjwm sna ciaimt xfm okug cyyfxis lqfki ga nyevuobgzy mzed vkj sffmg sslqlecjw rdhuqoztr yyck kpj ryuexkcpel sfphaguts eutxcadcr bswhufv badedphgsw mj zbgr xzyalzgmo nac dlwmo xfey ssg zdkgraz omxf bgkjd jzyalzgmo nau nna ih mxvxkxxctc gryk grxsh zkj qqgf rdpijialmo la ryqwbnjwg gwv kkecji sv zcsevty mzek jfljieoxket xfna xfyaa qw zxfji sfphagutn uycx ksabnjwg gjf rdbgjwj mqwiqse ryqwbnjwb wlwk k ybcou rceg gjf kjgzk k ybsxji az vhoggry mlwryxkhmn pvu s kfqxk xkjiwq ryqwbnjwz sivi im likxkalho adut hu idqcadac nvkmagho nvxkgqxkc eeky ls eutxcitqowd eepcpuwyk qkjcalqel nat lkjurho iyjfekalsdwd nat lkji ikzp izyalqosfon hu balwzefkkaanu tutxcadeepcpuag nac dkkk ifxjigmaljwi khueq na ryqwbnjw ylm sylchi gx fce c exwadlymxsx ryqwbnag epryo bjf jy xffk ryqwbnjw wy uksdsx wvivalgf sv na zcsevtji ggdm zcsevtji ompj zcsevti gh zo gxkjifkk k eepcpuag eprya zutxcadeepcpuag epryqf rdfoektc frgq bgkjm bemqjmo ompj pesrxc xfu ebgkaq tg tm lutxcada nyevumxsx sv xfr yvuxkzj wzqjr yoc ryqwbnjwf ejsuuon vu dgwmnd xkjiwq hu xfz ovuu kz forsvcr lgxk xfey zyk kyeryjifktc ivmpkxsvel hulqsnqcfxg gwv sssfdxa ql dkxlrhua yjffqdmyyu tutxcadcr ce copj ce ryqwbnjw ztceee qx xfs ase la ryqwbngqskmlalh wgsrni kedqoho onutmspimo ssj fpuqwo nutxcadt zivqay wq tdiji zsgmw iutxcadcr zyi gljdkgfm xk hnai ilrgrssaljwi keky ls eutxcadyjuw cornhui ofkoraz kxn sbemo ryqwbnjw carqyaiyedf ejsuualjw xfjitmp wcejiy qceaqac kayjeraljw azonxcvug gjf sfphagut gmequtkxsvmb huonkxxckacr hukntrhuo autxcadcr omxf bgkjm bbefkgqag sfphagutg gwv la huonwvlq hu epryqf owni ce niuewybni ijf idzgkkd jszecuw xfq wemybhk wzqji gfxjigmaljw ihqrwif xkj ilecalgf rdbgjwu izewha nsvcaf sekutu ujf xfx kcufkfqw iutxcadcr omxfemc ppjkyl qkj iyxqyazekaq tl dkj qra eqtdq ce fkgqe cjwrofkkaq tt pjysrwvgzmo hu xfd wadilphkacr la mlccalgf ryqwbnjwr cjiowz syazefqx kekuta gylsxmo xfn qhumxsx xkjiwq la na fkgqag aulque xfw islylsxa qm bylitsx la xfw iutxcadcr ihqrwif xgw vu ukwv fkgqe cl qkj eehu ssaca qm fkjcassl bjwmxsx hu vhzewyl lfkxwho nvbiyapikjj vagyltp ivcasx epmbx fbubgkjecqpt lkj iygzdws vg ty jjffqwyxaut wenaalwd la ryqwbnag nizyoa sv bsztywf xkj niueagho rdpijialmo la fkgqe citbw nvkmagwyu zaqchl mrmcrsvml lqjwey anqct lwhm sylchw iutxcady ja njf r f xfgqwma njf whl qkku kj zitalbw nvkmagwyet dmxkaleptc ze ce t mzyalzgi gm lutxcadcr ssa nkzp izyalqosfon hu idzgkkj vkkxckacr owni ce idzgaqaufxkjece cc rcpmb lgjip ukji iwv zmkya nj mput mzyalzgmo ryqwbnjwm sna ih rqkkont lxjemxs ihcp azssy of rjskagmalgf wybgpimo nam xv dmekjmlcho udkgrecjwv kvu hulqnaeq lgxk iyqaag kxkm xkonp hkj fkukzskjwv xfk kfxngzyalzga ql dkj ryqwbnjw wy sspjtc xfd wyazeady ja le cazfxagho sv xfw iutxcadm lsldwh zyla li bzyitmo ryqwbnagho nae hxfji nizyoaw ml dkj piilecon ecpc lgxk na ryqwbnjw iypjdmmo omxf t myazeadd tbgr xzyalzgmo ssd tk knnpkbifktc rdfoekhuu mkuee kxb uorsvcr

- Now input the above cipher text in line 17 of the code.

```
17    raw_text = input()
18    raw_text = raw_text.lower()
19    raw_text = "".join(c for c in raw_text if c in alphabets)
20    raw_text
```

The variable ***raw_input*** has the cipher string.

- Now, in the **decrypt_2** function, change the strings **a,** and **b** in line 152,153.
  a = “el”
  b = “th”
  As, the most common diagrams in the above plaintext are “el” and “th” respectively.

```
151    def decrypt_2(raw_cipher,n=2):
152        a = "th"   # change to "el"
153        b = "he"   # change to "th"
154        C = most_common_digrams(raw_cipher).T
155        P = make_two_key(a,b).T
156        cipher = prepare_text(raw_cipher,2)
157        temp = Matrix(P)
158        Pinv = temp.inv_mod(26)
159        key = np.mod(np.dot(C,Pinv),26).astype(int)  # encryption key is found
160        print("encryption key is : ",key)
```

- Call the function **decrypt_2** with the cipher string **raw_text** as input.

  Here you have to add the commands for printing the original plain text as it was not present, so substitute line 214 from the following piece of code to print.

```
214    original_plain_text = decrypt_2(raw_text,2)
215
216    print()
217    print("original text is : ")
218    print(original_plain_text)
```

- Correct key K is printed and It returns the correct plaintext as output.

- To run the submitted code (2016CS10347.py):
  1. Install Numpy using : pip install numpy
  2. Install sympy using : pip install sympy
  3. Open command prompt to the location of the code.
  4. Run the command : python 2016CS10347.py
  5. Paste the copied cipher-text from the above example, and press enter.
  6. The key and the plain text will be printed, with some other print results too.