LAB REPORT

SWE430: Information and Network Security

Submitted to

Partha Protim Paul
Lecturer
Dept. of Software Engineering
Shahjalal University of Science and Technology

Submitted by

Md. Mehrajul Islam
2019831074

LAB 2
Attacking Classic Crypto Systems

Checkpoint 1: Breaking Caesar Cipher

Given Cipher: *odroboewscdrolocdcwkbdmyxdbkmdzvkdpybwyeddrobo*

ceaserCracker.py

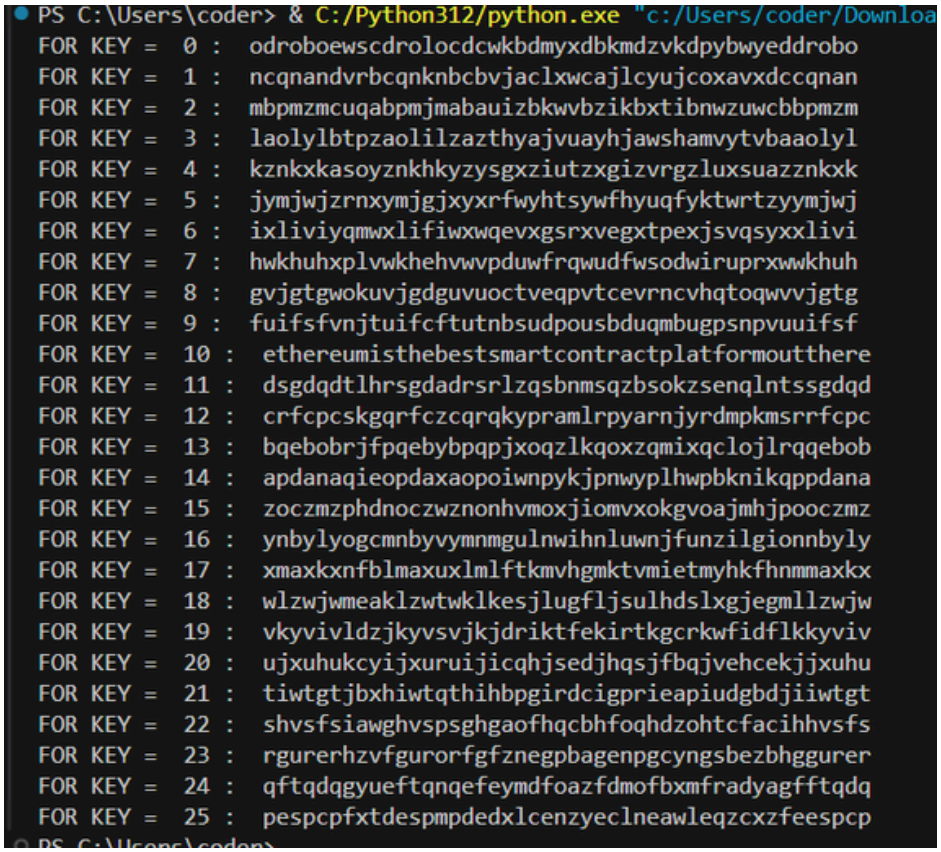
C: > Users > coder > Downloads > MEHRAJ > MEHRAJ > LAB 2 > TASK 1 > ceaserCracker.py

```python
cipherText = 'odroboewscdrolocdcwkbdmyxdbkmdzvkdpybwyeddrobo'
alphabets = 'abcdefghijklmnopqrstuvwxyz'

for key in range(len(alphabets)):

    decrypted = ''

    for ch in cipherText:

        if ch in alphabets:

            ascii = alphabets.find(ch)
            ascii = ascii - key

            if ascii < 0:
                ascii = ascii + len(alphabets)

            decrypted = decrypted + alphabets[ascii]

        else:
            decrypted = decrypted + ch

    print('FOR KEY = ', key, ': ', decrypted)
```

Algorithm:

- *There can be possible 25 keys. So first a loop with* ***key 0 to 25***
- *In each iteration for each character in ciphertext get its ASCII and deduct key value from that*
- *Add the character in the final string and print them after each iteration*
- *Lastly go through all of them and find the meaningful one*

OUTPUT



HERE FOR KEY = 10 WE GET:
ethereumisthebestsmartcontractplatformoutthere

SO DECRYPTED MESSAGE IS:
ethereum is the best smart contract platform out there

Checkpoint 2: Breaking Substitution Cipher

Cipher 1: *af p xpkcaqvnpk pfg, af ipqe qpri, gauuikifc tpw, ceiri udvk tiki afgarxifrphni cd eaowvmd popkwn, hiqpvri du ear jvaql vfgikrcpfgafm du cei xkafqaxnir du xrwqedearcdkw pfg du ear aopmafpcasi xkdhafmr afcd fit pkipr. ac tpr qdoudkcafm cd lfdt cepc au pfwceafm epxxifig cd ringdf eaorinu hiudki cei opceiopcaqr du cei uaing qdvng hi qdoxnicinw tdklig dvc pfg edt rndtnw ac xkdqiigig, pfg edt odvfcpafdvr cei dhrcpqnir--ceiki tdvng pc niprc kiopafdfi mddg oafg cepc tdvng qdfcafvi cei kiripkqe*

SUBSTITUTION CYPHER CODE

```cpp
map < char , int > cipherTextFrequency;
int numberOfReplacableCharecters = 0;

string ciphertext = "af p xpkcaqvnpk pfg, af ipqe qpri, gauuikifc tpw, ceiri udvk tiki
    afgarxifrphni cd eaowvmd popkwn, hiqpvri du ear jvaql vfgikrcpfgafm du cei xkafqaxnir du
    xrwqedearcdkw pfg du ear aopmafpcasi xkdhafmr afcd fit pkipr. ac tpr qdoudkcafm cd lfdt cepc
    au pfwceafm epxxifig cd ringdf eaorinu hiudki cei opceiopcaqr du cei uaing qdvng hi
    qdoxnicinw tdklig dvc pfg edt rndtnw ac xkdqiigig, pfg edt odvfcpafdvr cei dhrcpqnir--ceiki
    tdvng pc niprc kiopafdfi mddg oafg cepc tdvng qdfcafvi cei kiripkqe";

for(auto ch: ciphertext){
    if('a' <= ch && 'z' >= ch) cipherTextFrequency[ch]++; numberOfReplacableCharecters++;
}

vector < pair < double , char > > cipherFreqList;
string englishLanguageFreqOrder = "etaonhisrdluwmgcfybpkvjxzq";
string key = "abcdefghijklmnopqrstuvwxyz";
string ciphertextFrequencyOrder = "";

for(auto freqData: cipherTextFrequency) {
    cipherFreqList.push_back({(freqData.second * 100.00) / numberOfReplacableCharecters, freqData
        .first});
}

sort(cipherFreqList.begin(), cipherFreqList.end());

cout << "CIPHERTEXT FREQUENCY ORDER : " << endl;
for(auto freq: cipherFreqList){
    cout << freq.second << " >> " << freq.first << "% " << endl;
    ciphertextFrequencyOrder += freq.second;
}

reverse(ciphertextFrequencyOrder.begin(), ciphertextFrequencyOrder.end());

  cout << endl << "ciphertextFrequencyOrder : " << ciphertextFrequencyOrder << endl;
  cout << "englishLanguageFrequencyOrder : " << englishLanguageFreqOrder << endl << endl;

  int indx = 0;
  for(auto ch: ciphertextFrequencyOrder){
      cout << key[(int)englishLanguageFreqOrder[indx] - (int)'a'] << " will be replaced as " << ch
          << endl;
      key[(int)englishLanguageFreqOrder[indx] - (int)'a'] = ch;
      indx++;
  }

  cout << endl << " >>> KEY FOUND : " << key << endl << endl;

  string decrypted = ciphertext;

  int i = 0;
  for(auto lol: ciphertext) {
      if('a' <= ciphertext[i] && 'z' >= ciphertext[i]) decrypted[i] = key[(int)lol - (int)'a'];
      i++;
  }

  cout << " >>> AFTER DECRYPTION : " << endl;
  cout << decrypted << endl;

  return 0;
}
```

ALGORITHM FOR CRACKING SUBSTITUTION CIPHER

ALGORITHM:

- *Calculate the percentage of the frequency for the cipher text*
- *Sort them according to their appearance*
- *Compare them with the English language frequency*
- *Replace the cipher text according to the frequency of the cipher text and English language*
- *Check the result*

OUTPUT FOR CIPHER 1

CIPHERTEXT FREQUENCY ORDER :

j >> 0.203666%
s >> 0.203666%
l >> 0.610998%
h >> 1.222%
m >> 1.42566%
w >> 1.62933%
x >> 2.03666%
o >> 2.24033%
t >> 2.24033%
u >> 2.64766%
v >> 2.64766%
q >> 3.05499%
n >> 3.25866%
g >> 3.86965%
k >> 3.86965%
e >> 4.48065%
r >> 4.68432%
f >> 6.10998%
a >> 6.31365%
p >> 6.51731%
c >> 6.72098%
d >> 7.33198%
i >> 9.36864%

ciphertextFrequencyOrder :
idcpafrekgnqvutoxwmhlsj

englishLanguageFrequencyOrder :
etaonhisrdluwmgcfybpkvjxzq

KEY FOUND : cmogixtfrjlnuaphqkedqsvxwz

AFTER DECRYPTION :

cx h xhlocqsahl hxt, cx rhqi qhkr, tcqqrlrxo dhv, oirkr qgsl drlr cxtckxrxkhfar og icpvsug hphlva, frqhskr gq ick jscqn sxtrlkohxtcxu gq oir xlcxqcxark gq xkvqigickoglv hxt gq ick cphucxhocer xlgfcxuk cxog xrd hlrhk. co dhk qgpqglocxu og nxgd oiho cq hxvoicxu ihxxrxrt og kratgx icpkraq frqglr oir phoirphocqk gq oir qcrat qgsat fr qgpxarorav dglnrt gso hxt igd kagdav co xlgqrrtrt, hxt igd pgsxohcxgsk oir gfkohqark--oirlr dgsat ho arhko lrphcxgxr uggt pcxt oiho dgsat qgxocxsr oir lrkrhlqi

Cipher 2: *aceah toz puvg vcdl omj puvg yudqecov, omj loj auum klu thmjuv hs klu zlcvu shv zcbkg guovz, upuv zcmdu lcz vuwovroaeu jczoyyuovomdu omj qmubyudkuj vukqvm. klu vcdluz lu loj avhqnlk aodr svhw lcz kvopuez loj mht audhwu o ehdoe eunumj, omj ck toz yhyqeoveg auecupuj, tlokupuv klu hej sher wcnlk zog, klok klu lcee ok aon umj toz sqee hs kqmmuez zkqssuj tckl kvuozqvu. omj cs klok toz mhk umhqnl shv sowu, kluvu toz oezh lcz yvhehmnuj pcnhqv kh wovpue ok. kcwu thvu hm, aqk ck zuuwuj kh lopu eckkeu ussudk hm wv. aonncmz. ok mcmukg lu toz wqdl klu zowu oz ok scskg. ok mcmukg-mcmu klug aunom kh doee lcw tuee-yvuzuvpuj; aqk qmdlomnuj thqej lopu auum muovuv klu wovr. kluvu tuvu zhwu klok zlhhr klucv luojz omj klhqnlk klcz toz khh wqdl hs o nhhj klcmn; ck zuuwuj qmsocv klokomghmu zlhqej yhzzuzz (oyyovumkeg) yuvyukqoe ghqkl oz tuee oz (vuyqkujeg) cmubloqzkcaeu tuoekl. ck tcee lopu kh au yocj shv, klug zocj. ck czm'k mokqvoe, omj kvhqaeu tcee dhwu hs ck! aqk zh sov kvhqaeu loj mhk dhwu; omj oz wv. aonncmz toz numuvhqz tckl lcz whmug, whzk yuhyeu tuvu tceecmn kh shvncpu lcw lcz hjjckcuz omj lcz nhhj shvkqmu. lu vuwocmuj hm pczckcmn kuvwz tckl lcz vueokcpuz (ubduyk, hs dhqvzu, klu zodrpceeu- aonncmzuz), omj lu loj womg juphkuj ojwcvuvz owhmn klu lhaackz hs yhhv omj qmcwyhvkomk sowcecuz. aqk lu loj mh dehzu svcumjz, qmkce zhwu hs lcz ghqmnuv dhqzcmz aunom kh nvht qy. klu uejuzk hs kluzu, omj aceah'z sophqvcku, toz ghqmn svhjh aonncmz. tlum aceah toz mcmukg-mcmu lu ojhykuj svhjh oz lcz lucv, omj avhqnlk lcw kh ecpu ok aon umj; omj klu lhyuz hs klu zodrpceeu- aonncmzuz tuvu scmoeeg jozluj. aceah omj svhjh loyyumuj kh lopu klu zowu acvkljog, zuykuwauv 22mj. ghq loj aukkuv dhwu omj ecpu luvu, svhjh wg eoj, zocj aceah hmu jog; omj klum tu dom dueuavoku hqv acvkljog-yovkcuz dhwshvkoaeg khnukluv. ok klok kcwu svhjh toz zkcee cm lcz ktuumz, oz klu lhaackz doeeuj klu cvvuzyhmzcaeu ktumkcuz auktuum dlcejlhhj omj dhwcmn hs onu ok klcvkg-klvuu*

OUTPUT FOR CIPHER 2

CIPHERTEXT FREQUENCY ORDER :

b >> 0.205339%
r >> 0.359343%
p >> 1.12936%
g >> 1.43737%
y >> 1.43737%
d >> 1.48871%
t >> 1.74538%
n >> 1.89938%
s >> 1.95072%
w >> 1.95072%
q >> 2.15606%
a >> 2.41273%
e >> 3.64476%
j >> 3.79877%
v >> 4.36345%
m >> 4.8768%
z >> 4.8768%
l >> 4.97947%
c >> 5.23614%
h >> 5.80082%
o >> 6.72485%
k >> 6.77618%
u >> 10.1643%

ciphertextFrequencyOrder :
ukohclzmvjeaqwsntdygprb

englishLanguageFrequencyOrder :
etaonhisrdluwmgcfybpkvjxzq

KEY FOUND : oynjutslzbpewchgqvmkarqxdz

AFTER DECRYPTION :

onuol khz gars rnje hwb gars dajqunhr, hwb ehb oaaw pea klwbar lm pea zenra mlr znyps sahrz, **evar** znwja enz raqhrvhoua bnzhddahrhwja hwb qwaydajpab rapqrw. pea rnjeaz ea ehb orlqcep ohjv mrlq enz prhgauz ehb wlk oajlqa h uljhu uacawb, hwb np khz dldquhrus oaunagab, kehpagar pea lub mluv qncep zhs, pehp pea enuu hp ohc awb khz mquu lm pqwwauz zpqmmab knpe prahzqra. hwb nm pehp khz wlp awlqce mlr mhqa, peara khz huzl enz drlulwcab gnclqr pl qhrgau hp. pnqa klra lw, oqp np zaaqab pl ehga unppua ammajp lw qr. ohccnwz. hp wnwaps ea khz qqje pea zhqa hz hp mnmps. hp wnwaps-wnwa peas oachw pl jhuu enq kauu-drazargab; oqp qwjehwcab klqub ehga oaaw wahrar pea qhrv. peara kara zlqa pehp zellv peanr eahbz hwb pelqcep penz khz pll qqje lm h cllb penwc; np zaaqab qwmhnr pehphwslwa zelqub **plssass** (hddhrawpus) dardapqhu slqpe hz kauu hz (radqpabus) nwayehqzpnoua kahupe. np knuu ehga pl oa dhnb mlr, peas zhnb. np nzw'p whpqrhu, hwb prlqoua knuu jlqa lm np! oqp zl mhr prlqoua ehb wlp jlqa; hwb hz qr. ohccnwz khz cawarlqz knpe enz qlwas, qlzp daldua kara knuunwc pl mlrcnga enq enz lbbnpnaz hwb enz cllb mlrpqwa. ea raqhnwab lw gnznpnwc parqz knpe enz rauhpngaz (**expcdt**, lm jlqrza, pea zhjvgnuua- ohccnwzaz), hwb ea ehb qhws baglpab hbqnrarz hqlwc pea eloonpz lm dllr hwb qwnqdlrphwp mhqnunaz. oqp ea ehb wl julza mrnawbz, qwpnu zlqa lm enz slqwcar jlqznwz oachw pl crlk qd. pea aubazp lm peaza, hwb onuol'z mhglqrnpa, khz slqwc mrlbl ohccnwz. keaw onuol khz wnwaps-wnwa ea hbldpab mrlbl hz enz eanr, hwb orlqcep enq pl unga hp ohc awb; hwb pea eldaz lm pea zhjvgnuua- ohccnwzaz kara mnwhuus bhzeab. onuol hwb mrlbl ehddawab pl ehga pea zhqa onrpebhs, zadpaqoar 22wb. slq ehb oappar jlqa hwb unga eara, mrlbl qs uhb, zhnb onuol lwa bhs; hwb peaw ka jhw jauaorhpa lqr onrpebhs-dhrpnaz jlqmlrphous plcapear. hp pehp pnqa mrlbl khz zpnuu nw enz pkaawz, hz pea eloonpz jhuuab pea nrrazdlwznoua pkawpnaz oapkaaw jenubellb hwb jlqnwc lm hca hp penrps-peraa

CONCLUSION :

Here in the both cyphertext I tried to solve it using the English language frequency. But still it was unable to crack it. Because the text was too small. As we know that if the length of the ciphertext is bigger the breaking of substitution cipher becomes very easy. As the first ciphertext is smaller that is why it cannot be decoded at all but the second one is relatively bigger which makes it a little bit better suit for decoding. So, in the second one we can see some words like “evar”(ever), “plssass”(possess) and “expcdt”(expect).