Introduction to Information Security - CS 458 - Spring 2019
Lab 1 - Secret-Key Encryption
Due: Blackboard Sunday February 24th, 2019 by 11:59pm

2 Lab Tasks

2.1 Task 1: Frequency Analysis Against Monoalphabetic Substitution Cipher

Using Python

1. Most frequent English letters: e t a o i n s
2. 1-letter English words: a i
3. Most frequently doubled letters in English: s e t f l m o
4. Most frequent 2-letter words in English: an, at, as, he, be, in, is, it, on, or, to, of, do, go, no, so, my
5. Most frequent 3-letter words in English: the, and, for, was, his, not, but, you, are, her
6. Most frequent initial letters in English: t a s o i
7. Most frequent final letters in English: e s d n t

Plaintext Substitution:

1. Frequency analysis is done through Python and steps are identified.
2. Result:
   'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENI NG EVERY DAY A RIGOROUS EDUCATION COMBINING THE THEORY OF INFORMATION AND COM PUTATION WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE KEY TO SUCCESS AS ON E OF THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR TMENT AT IIT HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH QUALITY EDU CATION IN SMALL CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP ORTUNITIES IN INDUSTRY AND NATIONAL LABORATORIESIIT STUDENTS WORK WITH OUR FA CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIBUTED S YSTEMS INFORMATION RETRIEVAL COMPUTER NETWORKING INTELLIGENT INFORMATION SYST EMS AND ALGORITHMS THE DEPARTMENT OFFERS BACHELOR OF SCIENCE MASTER OF SCIENC E PROFESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAKE EVENING CLASSES AND LO NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN G AS AMONG THE BEST AT THE UNIVERSITY AND OUR FACULTY HAVE WON NUMEROUS TEACHING AWARDS THE SECRET SENTENCE IS GOOD JOB GUYS'

Ciphertext

In [1]:
```
from collections import Counter
file = open("ciphertext.txt","r")

ciphertext = file.read()

ciphertext = ciphertext.replace("\n","")

print(ciphertext)
```

```
hfcnkopw ahyplhp ya wznysgj hxzlvylv oxp qfwgs qyox lpq spdpgfncploa xznnplyl
v pdpwj szj z wyvfwfka pskhzoyfl hfceylylv oxp oxpfwj fu ylufwczoyfl zls hfcn
kozoyfl qyox xzlsafl ajaopca zls afuoqzwp spayvl ya oxp ipj of akhhpaa za flp
fu oxp fgspao hfcnkopw ahyplhp spnzwocploa yl oxp hxyhzvf zwpz oxp ha spnzwoc
plo zo yyo xza z gflv xyaofwj fu cppoylv oxya hxzggplvp oxwfkvx mkzgyoj pskhz
oyfl yl aczgg hgzaawffc pldywflcploa zgflv qyox ylopwlaxyn zls wpapzwhx fnnfw
oklyoypa yl ylskaowj zls lzoyflzg gzefwzofwypayyo aoksploa qfwi qyox fkw uzhk
goj fl qfwgshgzaa wpapzwhx yl zwpza oxzo ylhgksp szoz ahyplhp syaowyekops aja
opca ylufwczoyfl wpowypdzg hfcnkopw lpoqfwiylv ylopggyvplo ylufwczoyfl ajaopc
a zls zgvfwyoxca oxp spnzwocplo fuupwa ezhxpgfw fu ahyplhp czaopw fu ahyplhp
nwfupaayflzg czaopw zls nxs spvwppa ngka vwzskzop hpwoyuyhzopa zhhpgpwzops hf
kwapa zls lflspvwpp aoksj nzwooycp aoksploa hzl ozip pdplylv hgzaapa zls gflv
syaozlhp aoksploa hzl pzwl czaopwa spvwppa flgylp aoksploa wzop fkw opzhxylv
za zcflv oxp epao zo oxp klydpwayoj zls fkw uzhkgoj xzdp qfl lkcpwfka opzhxyl
v zqzwsaoxp aphwpo aploplhp ya vffs bfe vkja
```

Text frequencies

In [2]:
```
pt_frequency = [('e', '12.7 %'),('t', '9.1 %'),('a', '8.2 %'),('o', '7.5 %'),('i', '7.0 %'),('n', '6.7 %'),('s', '6.3 %'),('h', '6.1 %'),('r', '6.0 %'),('d', '4.3 %'),('l', '4.0 %'),('u', '2.8 %'),('c', '2.8 %'),('m', '2.4 %'),('w', '2.4 %'),('f', '2.2 %'),('y', '2.0 %'),('g', '2.0 %'),('p', '1.9 %'),('b', '1.5 %'),('v', '1.0 %'),('k', '0.8 %'),('x', '0.2 %'),('j', '0.2 %'),('q', '0.1 %'),('z', '0.1 %')]
ct_frequency = []
alphabets = 'abcdefghijklmnopqrstuvwxyz'
counter = 0
size = len(ciphertext.replace(" ",""))

for a in alphabets:
    counter = 0
    for c in ciphertext:
        if not  c == ' ':
            if a == c:
                counter += 1

    ct_frequency += [(str(a),float("%.2f" % ((counter/size)*100)))]

ct_frequency.sort(key = lambda x: x[1], reverse = True)
ct_frequency = [(key, str(value) + ' %') for key, value in ct_frequency]
print('Plaintext Frequency:\n',pt_frequency)
print('\n')
print('Ciphertext Frequency:\n',ct_frequency)
```

```
Plaintext Frequency:
 [('e', '12.7 %'), ('t', '9.1 %'), ('a', '8.2 %'), ('o', '7.5 %'), ('i', '7.0 %'), ('n', '6.7 %'), ('s', '6.3 %'), ('h', '6.1 %'), ('r', '6.0 %'), ('d', '4.3 %'), ('l', '4.0 %'), ('u', '2.8 %'), ('c', '2.8 %'), ('m', '2.4 %'), ('w', '2.4 %'), ('f', '2.2 %'), ('y', '2.0 %'), ('g', '2.0 %'), ('p', '1.9 %'), ('b', '1.5 %'), ('v', '1.0 %'), ('k', '0.8 %'), ('x', '0.2 %'), ('j', '0.2 %'), ('q', '0.1 %'), ('z', '0.1 %')]


Ciphertext Frequency:
 [('p', '12.07 %'), ('o', '9.68 %'), ('l', '8.84 %'), ('a', '8.43 %'), ('z', '8.22 %'), ('y', '7.08 %'), ('f', '6.66 %'), ('w', '6.35 %'), ('h', '4.68 %'), ('s', '4.27 %'), ('x', '3.64 %'), ('k', '3.23 %'), ('g', '3.12 %'), ('c', '2.71 %'), ('v', '2.71 %'), ('n', '1.87 %'), ('j', '1.66 %'), ('u', '1.56 %'), ('q', '1.25 %'), ('d', '0.73 %'), ('e', '0.62 %'), ('i', '0.42 %'), ('b', '0.1 %'), ('m', '0.1 %'), ('r', '0.0 %'), ('t', '0.0 %')]
```

One letter word in ciphertext

In [3]:
```
one_letter = []
words = ciphertext.split(" ")
for w in words:
    if len(w) == 1:
        one_letter += [w]

print(Counter(one_letter))
```

```
Counter({'z': 2})
```

Double letters

In [4]:
```
double_letter = []
for w in words:
    temp = ""
    for l in w:
        if temp == l:
            double_letter += [l]
        temp = l
print(Counter(double_letter))
```

```
Counter({'a': 5, 'p': 4, 'g': 3, 'n': 2, 'h': 2, 'y': 2, 'f': 2, 'u': 1, 'o':
1})
```

Two letter word in ciphertext

In [5]:
```
two_letter = []
for w in words:
    if len(w) == 2:
        two_letter += [w]
print(Counter(two_letter))
```

```
Counter({'fu': 5, 'yl': 4, 'ya': 3, 'za': 2, 'zo': 2, 'of': 1, 'ha': 1, 'fl':
1})
```

Three letter word in ciphertext

In [6]:
```
three_letter = []
for w in words:
    if len(w) == 3:
        three_letter += [w]
print(Counter(three_letter))
```

```
Counter({'oxp': 9, 'zls': 9, 'fkw': 3, 'hzl': 2, 'lpq': 1, 'szj': 1, 'ipj':
1, 'flp': 1, 'yyo': 1, 'xza': 1, 'nxs': 1, 'qfl': 1, 'bfe': 1})
```

Frequent initial letters

In [7]:
```
size = 0
initial_frequency = []
for a in alphabets:
    counter = 0
    for w in words:
        if len(w) > 1:
            if w[0] == a:
                counter += 1
    size += counter

    initial_frequency += [(str(a),float("%.2f" % counter))]

initial_frequency.sort(key = lambda x: x[1], reverse = True)
initial_frequency = [(key, str("%.2f" %(value/size)) + ' %') for key, value in
initial_frequency]

print(initial_frequency)
```

```
[('z', '0.12 %'), ('a', '0.11 %'), ('o', '0.11 %'), ('h', '0.09 %'), ('y',
'0.09 %'), ('f', '0.09 %'), ('s', '0.06 %'), ('q', '0.05 %'), ('p', '0.04
%'), ('w', '0.04 %'), ('l', '0.03 %'), ('x', '0.03 %'), ('c', '0.02 %'),
('n', '0.02 %'), ('g', '0.02 %'), ('v', '0.02 %'), ('e', '0.01 %'), ('u', '0.
01 %'), ('b', '0.01 %'), ('i', '0.01 %'), ('k', '0.01 %'), ('m', '0.01 %'),
('d', '0.00 %'), ('j', '0.00 %'), ('r', '0.00 %'), ('t', '0.00 %')]
```

Frequent final letters

In [8]:
```
size = 0
final_frequency = []
for a in alphabets:
    counter = 0
    for w in words:
        if len(w) > 1:
            if w[-1] == a:
                counter += 1
    size += counter

    final_frequency += [(str(a),float("%.2f" % counter))]

final_frequency.sort(key = lambda x: x[1], reverse = True)
final_frequency = [(key, str("%.2f" %(value/size)) + ' %') for key, value in f
inal_frequency]

print(final_frequency)
```

```
[('a', '0.21 %'), ('p', '0.17 %'), ('l', '0.11 %'), ('s', '0.09 %'), ('j',
'0.07 %'), ('o', '0.07 %'), ('v', '0.07 %'), ('w', '0.06 %'), ('x', '0.04
%'), ('u', '0.03 %'), ('g', '0.02 %'), ('f', '0.01 %'), ('z', '0.01 %'),
('c', '0.01 %'), ('e', '0.01 %'), ('i', '0.01 %'), ('n', '0.01 %'), ('q', '0.
01 %'), ('b', '0.00 %'), ('d', '0.00 %'), ('h', '0.00 %'), ('k', '0.00 %'),
('m', '0.00 %'), ('r', '0.00 %'), ('t', '0.00 %'), ('y', '0.00 %')]
```

Substitution

In [9]:
```
cipher_to_plain = {}
```

One Letter Frequency (z)

In [10]:
```
cipher_to_plain['z'] = 'A'
ciphertext = ciphertext.replace('z','A')
ciphertext
```

Out[10]:
```
'hfcnkopw ahyplhp ya wAnysgj hxAlvylv oxp qfwgs qyox lpq spdpgfncploa xAnnply
lv pdpwj sAj A wyvfwfka pskhAoyfl hfceylylv oxp oxpfwj fu ylufwcAoyfl Als hfc
nkoAoyfl qyox xAlsafl ajaopca Als afuoqAwp spayvl ya oxp ipj of akhhpaa Aa fl
p fu oxp fgspao hfcnkopw ahyplhp spnAwocploa yl oxp hxyhAvf AwpA oxp ha spnAw
ocplo Ao yyo xAa A gflv xyaofwj fu cppoylv oxya hxAggplvp oxwfkvx mkAgyoj psk
hAoyfl yl acAgg hgAaawffc pldywflcploa Agflv qyox ylopwlaxyn Als wpapAwhx fnn
fwoklyoypa yl ylskaowj Als lAoyflAg gAefwAofwypayyo aoksploa qfwi qyox fkw uA
hkgoj fl qfwgshgAaa wpapAwhx yl AwpAa oxAo ylhgksp sAoA ahyplhp syaowyekops a
jaopca ylufwcAoyfl wpowypdAg hfcnkopw lpoqfwiylv ylopggyvplo ylufwcAoyfl ajao
pca Als Agvfwyoxca oxp spnAwocplo fuupwa eAhxpgfw fu ahyplhp cAaopw fu ahyplh
p nwfupaayflAg cAaopw Als nxs spvwppa ngka vwAskAop hpwoyuyhAopa AhhpgpwAops
hfkwapa Als lflspvwpp aoksj nAwooycp aoksploa hAl oAip pdplylv hgAaapa Als gf
lvsyaoAlhp aoksploa hAl pAwl cAaopwa spvwppa flgylp aoksploa wAop fkw opAhxyl
v Aa Acflv oxp epao Ao oxp klydpwayoj Als fkw uAhkgoj xAdp qfl lkcpwfka opAhx
ylv AqAwsaoxp aphwpo aploplhp ya vffs bfe vkja'
```

Double Letter + Final Letter Frequency (aa)

In [11]:
```
cipher_to_plain['a'] = 'S'
ciphertext = ciphertext.replace('a','S')
ciphertext
```

Out[11]:
```
'hfcnkopw Shyplhp yS wAnysgj hxAlvylv oxp qfwgs qyox lpq spdpgfncploS xAnnply
lv pdpwj sAj A wyvfwfkS pskhAoyfl hfceylylv oxp oxpfwj fu ylufwcAoyfl Als hfc
nkoAoyfl qyox xAlsSfl SjSopcS Als SfuoqAwp spSyvl yS oxp ipj of SkhhpSS AS fl
p fu oxp fgspSo hfcnkopw Shyplhp spnAwocploS yl oxp hxyhAvf AwpA oxp hS spnAw
ocplo Ao yyo xAS A gflv xySofwj fu cppoylv oxyS hxAggplvp oxwfkvx mkAgyoj psk
hAoyfl yl ScAgg hgASSwffc pldywflcploS Agflv qyox ylopwlSxyn Als wpSpAwhx fnn
fwoklyoypS yl ylskSowj Als lAoyflAg gAefwAofwypSyyo SoksploS qfwi qyox fkw uA
hkgoj fl qfwgshgASS wpSpAwhx yl AwpAS oxAo ylhgksp sAoA Shyplhp sySowyekops S
jSopcS ylufwcAoyfl wpowypdAg hfcnkopw lpoqfwiylv ylopggyvplo ylufwcAoyfl SjSo
pcS Als AgvfwyoxcS oxp spnAwocplo fuupwS eAhxpgfw fu Shyplhp cASopw fu Shyplh
p nwfupSSyflAg cASopw Als nxs spvwppS ngkS vwAskAop hpwoyuyhAopS AhhpgpwAops
hfkwSpS Als lflspvwpp Soksj nAwooycp SoksploS hAl oAip pdplylv hgASSpS Als gf
lvsySoAlhp SoksploS hAl pAwl cASopwS spvwppS flgylp SoksploS wAop fkw opAhxyl
v AS Acflv oxp epSo Ao oxp klydpwSyoj Als fkw uAhkgoj xAdp qfl lkcpwfkS opAhx
ylv AqAwsSoxp Sphwpo Sploplhp yS vffs bfe vkjS'
```

Most Frequent Letter (p)

In [12]:
```
cipher_to_plain['p'] = 'E'
ciphertext = ciphertext.replace('p','E')
ciphertext
```

Out[12]:
```
'hfcnkoEw ShyElhE yS wAnysgj hxAlvylv oxE qfwgs qyox lEq sEdEgfncEloS xAnnEly
lv EdEwj sAj A wyvfwfkS EskhAoyfl hfceylylv oxE oxEfwj fu ylufwcAoyfl Als hfc
nkoAoyfl qyox xAlsSfl SjSoEcS Als SfuoqAwE sESyvl yS oxE iEj of SkhhESS AS fl
E fu oxE fgsESo hfcnkoEw ShyElhE sEnAwocEloS yl oxE hxyhAvf AwEA oxE hS sEnAw
ocElo Ao yyo xAS A gflv xySofwj fu cEEoylv oxyS hxAggElvE oxwfkvx mkAgyoj Esk
hAoyfl yl ScAgg hgASSwffc EldywflcEloS Agflv qyox yloEwlSxyn Als wESEAwhx fnn
fwoklyoyES yl ylskSowj Als lAoyflAg gAefwAofwyESyyo SoksEloS qfwi qyox fkw uA
hkgoj fl qfwgshgASS wESEAwhx yl AwEAS oxAo ylhgksE sAoA ShyElhE sySowyekoEs S
jSoEcS ylufwcAoyfl wEowyEdAg hfcnkoEw lEoqfwiylv yloEggyvElo ylufwcAoyfl SjSo
EcS Als AgvfwyoxcS oxE sEnAwocElo fuuEwS eAhxEgfw fu ShyElhE cASoEw fu ShyElh
E nwfuESSyflAg cASoEw Als nxs sEvwEES ngkS vwAskAoE hEwoyuyhAoES AhhEgEwAoEs
hfkwSES Als lflsEvwEE Soksj nAwooycE SoksEloS hAl oAiE EdElylv hgASSES Als gf
lvsySoAlhE SoksEloS hAl EAwl cASoEwS sEvwEES flgylE SoksEloS wAoE fkw oEAhxyl
v AS Acflv oxE eESo Ao oxE klydEwSyoj Als fkw uAhkgoj xAdE qfl lkcEwfkS oEAhx
ylv AqAwsSoxE SEhwEo SEloElhE yS vffs bfe vkjS'
```

Two Word Frequency (at)

In [13]:
```
cipher_to_plain['o'] = 'T'
ciphertext = ciphertext.replace('o','T')
ciphertext
```

Out[13]:
```
'hfcnkTEw ShyElhE yS wAnysgj hxAlvylv TxE qfwgs qyTx lEq sEdEgfncElTS xAnnEly
lv EdEwj sAj A wyvfwfkS EskhATyfl hfceylylv TxE TxEfwj fu ylufwcATyfl Als hfc
nkTATyfl qyTx xAlsSfl SjSTEcS Als SfuTqAwE sESyvl yS TxE iEj Tf SkhhESS AS fl
E fu TxE fgsEST hfcnkTEw ShyElhE sEnAwTcElTS yl TxE hxyhAvf AwEA TxE hS sEnAw
TcElT AT yyT xAS A gflv xySTfwj fu cEETylv TxyS hxAggElvE Txwfkvx mkAgyTj Esk
hATyfl yl ScAgg hgASSwffc EldywflcElTS Agflv qyTx ylTEwlSxyn Als wESEAwhx fnn
fwTklyTyES yl ylskSTwj Als lATyflAg gAefwATfwyESyyT STksElTS qfwi qyTx fkw uA
hkgTj fl qfwgshgASS wESEAwhx yl AwEAS TxAT ylhgksE sATA ShyElhE sySTwyekTEs S
jSTEcS ylufwcATyfl wETwyEdAg hfcnkTEw lETqfwiylv ylTEggyvElT ylufwcATyfl SjST
EcS Als AgvfwyTxcS TxE sEnAwTcElT fuuEwS eAhxEgfw fu ShyElhE cASTEw fu ShyElh
E nwfuESSyflAg cASTEw Als nxs sEvwEES ngkS vwAskATE hEwTyuyhATES AhhEgEwATEs
hfkwSES Als lflsEvwEE STksj nAwTTycE STksElTS hAl TAiE EdElylv hgASSES Als gf
lvsySTAlhE STksElTS hAl EAwl cASTEwS sEvwEES flgylE STksElTS wATE fkw TEAhxyl
v AS Acflv TxE eEST AT TxE klydEwSyTj Als fkw uAhkgTj xAdE qfl lkcEwfkS TEAhx
ylv AqAwsSTxE SEhwET SElTElhE yS vffs bfe vkjS'
```

Three Word Frequency (the)

In [14]:
```
cipher_to_plain['x'] = 'H'
ciphertext = ciphertext.replace('x','H')
ciphertext
```

Out[14]:
```
'hfcnkTEw ShyElhE yS wAnysgj hHAlvylv THE qfwgs qyTH lEq sEdEgfncElTS HAnnEly
lv EdEwj sAj A wyvfwfkS EskhATyfl hfceylylv THE THEfwj fu ylufwcATyfl Als hfc
nkTATyfl qyTH HAlsSfl SjSTEcS Als SfuTqAwE sESyvl yS THE iEj Tf SkhhESS AS fl
E fu THE fgsEST hfcnkTEw ShyElhE sEnAwTcElTS yl THE hHyhAvf AwEA THE hS sEnAw
TcElT AT yyT HAS A gflv HySTfwj fu cEETylv THyS hHAggElvE THwfkvH mkAgyTj Esk
hATyfl yl ScAgg hgASSwffc EldywflcElTS Agflv qyTH ylTEwlSHyn Als wESEAwhH fnn
fwTklyTyES yl ylskSTwj Als lATyflAg gAefwATfwyESyyT STksElTS qfwi qyTH fkw uA
hkgTj fl qfwgshgASS wESEAwhH yl AwEAS THAT ylhgksE sATA ShyElhE sySTwyekTEs S
jSTEcS ylufwcATyfl wETwyEdAg hfcnkTEw lETqfwiylv ylTEggyvElT ylufwcATyfl SjST
EcS Als AgvfwyTHcS THE sEnAwTcElT fuuEwS eAhHEgfw fu ShyElhE cASTEw fu ShyElh
E nwfuESSyflAg cASTEw Als nHs sEvwEES ngkS vwAskATE hEwTyuyhATES AhhEgEwATEs
hfkwSES Als lflsEvwEE STksj nAwTTycE STksElTS hAl TAiE EdElylv hgASSES Als gf
lvsySTAlhE STksElTS hAl EAwl cASTEwS sEvwEES flgylE STksElTS wATE fkw TEAhHyl
v AS Acflv THE eEST AT THE klydEwSyTj Als fkw uAhkgTj HAdE qfl lkcEwfkS TEAhH
ylv AqAwsSTHE SEhwET SElTElhE yS vffs bfe vkjS'
```

'SECRET' Substitution

In [15]:
```
cipher_to_plain['h'] = 'C'
cipher_to_plain['w'] = 'R'
ciphertext = ciphertext.replace('h','C')
ciphertext = ciphertext.replace('w','R')
ciphertext
```

Out[15]:
```
'CfcnkTER SCyElCE yS RAnysgj CHAlvylv THE qfRgs qyTH lEq sEdEgfncElTS HAnnEly
lv EdERj sAj A RyvfRfkS EskCATyfl Cfceylylv THE THEfRj fu ylufRcATyfl Als Cfc
nkTATyfl qyTH HAlsSfl SjSTEcS Als SfuTqARE sESyvl yS THE iEj Tf SkCCESS AS fl
E fu THE fgsEST CfcnkTER SCyElCE sEnARTcElTS yl THE CHyCAvf AREA THE CS sEnAR
TcElT AT yyT HAS A gflv HySTfRj fu cEETylv THyS CHAggElvE THRfkvH mkAgyTj Esk
CATyfl yl ScAgg CgASSRffc EldyRflcElTS Agflv qyTH ylTERlSHyn Als RESEARCH fnn
fRTklyTyES yl ylskSTRj Als lATyflAg gAefRATfRyESyyT STksElTS qfRi qyTH fkR uA
CkgTj fl qfRgsCgASS RESEARCH yl AREAS THAT ylCgksE sATA SCyElCE sySTRyekTEs S
jSTEcS ylufRcATyfl RETRyEdAg CfcnkTER lETqfRiylv ylTEggyvElT ylufRcATyfl SjST
EcS Als AgvfRyTHcS THE sEnARTcElT fuuERS eACHEgfR fu SCyElCE cASTER fu SCyElC
E nRfuESSyflAg cASTER Als nHs sEvREES ngkS vRAskATE CERTyuyCATES ACCEgERATEs
CfkRSES Als lflsEvREE STksj nARTTycE STksElTS CAl TAiE EdElylv CgASSES Als gf
lvsySTAlCE STksElTS CAl EARl cASTERS sEvREES flgylE STksElTS RATE fkR TEACHyl
v AS Acflv THE eEST AT THE klydERSyTj Als fkR uACkgTj HAdE qfl lkcERfkS TEACH
ylv AqARsSTHE SECRET SElTElCE yS vffs bfe vkjS'
```

'TEACHING' Substitution

In [16]:
```
cipher_to_plain['y'] = 'I'
cipher_to_plain['l'] = 'N'
cipher_to_plain['v'] = 'G'
ciphertext = ciphertext.replace('y','I')
ciphertext = ciphertext.replace('l','N')
ciphertext = ciphertext.replace('v','G')
ciphertext
```

Out[16]:
```
'CfcnkTER SCIENCE IS RAnIsgj CHANGING THE qfRgs qITH NEq sEdEgfncENTS HAnnENI
NG EdERj sAj A RIGfRfkS EskCATIfN CfceINING THE THEfRj fu INufRcATIfN ANs Cfc
nkTATIfN qITH HANsSfN SjSTEcS ANs SfuTqARE sESIGN IS THE iEj Tf SkCCESS AS fN
E fu THE fgsEST CfcnkTER SCIENCE sEnARTcENTS IN THE CHICAGf AREA THE CS sEnAR
TcENT AT IIT HAS A gfNG HISTfRj fu cEETING THIS CHAggENGE THRfkGH mkAgITj Esk
CATIfN IN ScAgg CgASSRffc ENdIRfNcENTS AgfNG qITH INTERNSHIn ANs RESEARCH fnn
fRTkNITIES IN INskSTRj ANs NATIfNAg gAefRATfRIESIIT STksENTS qfRi qITH fkR uA
CkgTj fN qfRgsCgASS RESEARCH IN AREAS THAT INCgksE sATA SCIENCE sISTRIekTEs S
jSTEcS INufRcATIfN RETRIEdAg CfcnkTER NETqfRiING INTEggIGENT INufRcATIfN SjST
EcS ANs AgGfRITHcS THE sEnARTcENT fuuERS eACHEgfR fu SCIENCE cASTER fu SCIENC
E nRfuESSIfNAg cASTER ANs nHs sEGREES ngkS GRAskATE CERTIuICATES ACCEgERATEs
CfkRSES ANs NfNsEGREE STksj nARTTIcE STksENTS CAN TAiE EdENING CgASSES ANs gf
NGsISTANCE STksENTS CAN EARN cASTERS sEGREES fNgINE STksENTS RATE fkR TEACHIN
G AS AcfNG THE eEST AT THE kNIdERSITj ANs fkR uACkgTj HAdE qfN NkcERfkS TEACH
ING AqARsSTHE SECRET SENTENCE IS Gffs bfe GkjS'
```

'TEACHING' Substitution

In [17]:
```
cipher_to_plain['f'] = 'O'
cipher_to_plain['c'] = 'M'
cipher_to_plain['n'] = 'P'
cipher_to_plain['k'] = 'U'
ciphertext = ciphertext.replace('f','O')
ciphertext = ciphertext.replace('c','M')
ciphertext = ciphertext.replace('n','P')
ciphertext = ciphertext.replace('k','U')
ciphertext
```

Out[17]:
```
'COMPUTER SCIENCE IS RAPIsgj CHANGING THE qORgs qITH NEq sEdEgOPMENTS HAPPENI
NG EdERj sAj A RIGOROUS EsUCATION COMeINING THE THEORj Ou INuORMATION ANs COM
PUTATION qITH HANsSON SjSTEMS ANs SOuTqARE sESIGN IS THE iEj TO SUCCESS AS ON
E Ou THE OgsEST COMPUTER SCIENCE sEPARTMENTS IN THE CHICAGO AREA THE CS sEPAR
TMENT AT IIT HAS A gONG HISTORj Ou MEETING THIS CHAggENGE THROUGH mUAgITj EsU
CATION IN SMAgg CgASSROOM ENdIRONMENTS AgONG qITH INTERNSHIP ANs RESEARCH OPP
ORTUNITIES IN INsUSTRj ANs NATIONAg gAeORATORIESIIT STUsENTS qORi qITH OUR uA
CUgTj ON qORgsCgASS RESEARCH IN AREAS THAT INCgUsE sATA SCIENCE sISTRIeUTEs S
jSTEMS INuORMATION RETRIEdAg COMPUTER NETqORiING INTEggIGENT INuORMATION SjST
EMS ANs AgGORITHMS THE sEPARTMENT OuuERS eACHEgOR Ou SCIENCE MASTER Ou SCIENC
E PROuESSIONAg MASTER ANs PHs sEGREES PgUS GRAsUATE CERTIuICATES ACCEgERATEs
COURSES ANs NONsEGREE STUsj PARTTIME STUsENTS CAN TAiE EdENING CgASSES ANs gO
NGsISTANCE STUsENTS CAN EARN MASTERS sEGREES ONgINE STUsENTS RATE OUR TEACHIN
G AS AMONG THE eEST AT THE UNIdERSITj ANs OUR uACUgTj HAdE qON NUMEROUS TEACH
ING AqARsSTHE SECRET SENTENCE IS GOOs bOe GUjS'
```

'RAPIDLY' Substitution

In [18]:
```
cipher_to_plain['s'] = 'D'
cipher_to_plain['g'] = 'L'
cipher_to_plain['j'] = 'Y'
ciphertext = ciphertext.replace('s','D')
ciphertext = ciphertext.replace('g','L')
ciphertext = ciphertext.replace('j','Y')
ciphertext
```

Out[18]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE qORLD qITH NEq DEdELOPMENTS HAPPENI
NG EdERY DAY A RIGOROUS EDUCATION COMeINING THE THEORY Ou INuORMATION AND COM
PUTATION qITH HANDSON SYSTEMS AND SOuTqARE DESIGN IS THE iEY TO SUCCESS AS ON
E Ou THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY Ou MEETING THIS CHALLENGE THROUGH mUALITY EDU
CATION IN SMALL CLASSROOM ENdIRONMENTS ALONG qITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LAeORATORIESIIT STUDENTS qORi qITH OUR uA
CULTY ON qORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIeUTED S
YSTEMS INuORMATION RETRIEdAL COMPUTER NETqORiING INTELLIGENT INuORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OuuERS eACHELOR Ou SCIENCE MASTER Ou SCIENC
E PROuESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIuICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAiE EdENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE eEST AT THE UNIdERSITY AND OUR uACULTY HAdE qON NUMEROUS TEACH
ING AqARDSTHE SECRET SENTENCE IS GOOD bOe GUYS'
```

'WORLD' Substitution

In [19]:
```
cipher_to_plain['q'] = 'W'
ciphertext = ciphertext.replace('q','W')
ciphertext
```

Out[19]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEdELOPMENTS HAPPENI
NG EdERY DAY A RIGOROUS EDUCATION COMeINING THE THEORY Ou INuORMATION AND COM
PUTATION WITH HANDSON SYSTEMS AND SOuTWARE DESIGN IS THE iEY TO SUCCESS AS ON
E Ou THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY Ou MEETING THIS CHALLENGE THROUGH mUALITY EDU
CATION IN SMALL CLASSROOM ENdIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LAeORATORIESIIT STUDENTS WORi WITH OUR uA
CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIeUTED S
YSTEMS INuORMATION RETRIEdAL COMPUTER NETWORiING INTELLIGENT INuORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OuuERS eACHELOR Ou SCIENCE MASTER Ou SCIENC
E PROuESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIuICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAiE EdENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE eEST AT THE UNIdERSITY AND OUR uACULTY HAdE WON NUMEROUS TEACH
ING AWARDSTHE SECRET SENTENCE IS GOOD bOe GUYS'
```

'EVERY' Substitution

In [20]:
```
cipher_to_plain['d'] = 'V'
ciphertext = ciphertext.replace('d','V')
ciphertext
```

Out[20]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENI
NG EVERY DAY A RIGOROUS EDUCATION COMeINING THE THEORY Ou INuORMATION AND COM
PUTATION WITH HANDSON SYSTEMS AND SOuTWARE DESIGN IS THE iEY TO SUCCESS AS ON
E Ou THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY Ou MEETING THIS CHALLENGE THROUGH mUALITY EDU
CATION IN SMALL CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LAeORATORIESIIT STUDENTS WORi WITH OUR uA
CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIeUTED S
YSTEMS INuORMATION RETRIEVAL COMPUTER NETWORiING INTELLIGENT INuORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OuuERS eACHELOR Ou SCIENCE MASTER Ou SCIENC
E PROuESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIuICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAiE EVENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE eEST AT THE UNIVERSITY AND OUR uACULTY HAVE WON NUMEROUS TEACH
ING AWARDSTHE SECRET SENTENCE IS GOOD bOe GUYS'
```

'FACULTY' Substitution

In [21]:
```
cipher_to_plain['u'] = 'F'
ciphertext = ciphertext.replace('u','F')
ciphertext
```

Out[21]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENI
NG EVERY DAY A RIGOROUS EDUCATION COMeINING THE THEORY OF INFORMATION AND COM
PUTATION WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE iEY TO SUCCESS AS ON
E OF THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH mUALITY EDU
CATION IN SMALL CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LAeORATORIESIIT STUDENTS WORi WITH OUR FA
CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIeUTED S
YSTEMS INFORMATION RETRIEVAL COMPUTER NETWORiING INTELLIGENT INFORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OFFERS eACHELOR OF SCIENCE MASTER OF SCIENC
E PROFESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAiE EVENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE eEST AT THE UNIVERSITY AND OUR FACULTY HAVE WON NUMEROUS TEACH
ING AWARDSTHE SECRET SENTENCE IS GOOD bOe GUYS'
```

'TAKE' Substitution

In [22]:
```
cipher_to_plain['i'] = 'K'
ciphertext = ciphertext.replace('i','K')
ciphertext
```

Out[22]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENI
NG EVERY DAY A RIGOROUS EDUCATION COMeINING THE THEORY OF INFORMATION AND COM
PUTATION WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE KEY TO SUCCESS AS ON
E OF THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH mUALITY EDU
CATION IN SMALL CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LAeORATORIESIIT STUDENTS WORK WITH OUR FA
CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIeUTED S
YSTEMS INFORMATION RETRIEVAL COMPUTER NETWORKING INTELLIGENT INFORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OFFERS eACHELOR OF SCIENCE MASTER OF SCIENC
E PROFESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAKE EVENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE eEST AT THE UNIVERSITY AND OUR FACULTY HAVE WON NUMEROUS TEACH
ING AWARDSTHE SECRET SENTENCE IS GOOD bOe GUYS'
```

'QUALITY' Substitution

In [23]:
```
cipher_to_plain['m'] = 'Q'
ciphertext = ciphertext.replace('m','Q')
ciphertext
```

Out[23]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENI
NG EVERY DAY A RIGOROUS EDUCATION COMeINING THE THEORY OF INFORMATION AND COM
PUTATION WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE KEY TO SUCCESS AS ON
E OF THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH QUALITY EDU
CATION IN SMALL CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LAeORATORIESIIT STUDENTS WORK WITH OUR FA
CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIeUTED S
YSTEMS INFORMATION RETRIEVAL COMPUTER NETWORKING INTELLIGENT INFORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OFFERS eACHELOR OF SCIENCE MASTER OF SCIENC
E PROFESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAKE EVENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE eEST AT THE UNIVERSITY AND OUR FACULTY HAVE WON NUMEROUS TEACH
ING AWARDSTHE SECRET SENTENCE IS GOOD bOe GUYS'
```

'BEST' Substitution

In [24]:
```
cipher_to_plain['e'] = 'B'
ciphertext = ciphertext.replace('e','B')
ciphertext
```

Out[24]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENI
NG EVERY DAY A RIGOROUS EDUCATION COMBINING THE THEORY OF INFORMATION AND COM
PUTATION WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE KEY TO SUCCESS AS ON
E OF THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH QUALITY EDU
CATION IN SMALL CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LABORATORIESIIT STUDENTS WORK WITH OUR FA
CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIBUTED S
YSTEMS INFORMATION RETRIEVAL COMPUTER NETWORKING INTELLIGENT INFORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OFFERS BACHELOR OF SCIENCE MASTER OF SCIENC
E PROFESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAKE EVENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE BEST AT THE UNIVERSITY AND OUR FACULTY HAVE WON NUMEROUS TEACH
ING AWARDSTHE SECRET SENTENCE IS GOOD bOB GUYS'
```

'JOB' Substitution

In [25]:
```
cipher_to_plain['b'] = 'J'
ciphertext = ciphertext.replace('b','J')
ciphertext
```

Out[25]:
```
'COMPUTER SCIENCE IS RAPIDLY CHANGING THE WORLD WITH NEW DEVELOPMENTS HAPPENI
NG EVERY DAY A RIGOROUS EDUCATION COMBINING THE THEORY OF INFORMATION AND COM
PUTATION WITH HANDSON SYSTEMS AND SOFTWARE DESIGN IS THE KEY TO SUCCESS AS ON
E OF THE OLDEST COMPUTER SCIENCE DEPARTMENTS IN THE CHICAGO AREA THE CS DEPAR
TMENT AT IIT HAS A LONG HISTORY OF MEETING THIS CHALLENGE THROUGH QUALITY EDU
CATION IN SMALL CLASSROOM ENVIRONMENTS ALONG WITH INTERNSHIP AND RESEARCH OPP
ORTUNITIES IN INDUSTRY AND NATIONAL LABORATORIESIIT STUDENTS WORK WITH OUR FA
CULTY ON WORLDCLASS RESEARCH IN AREAS THAT INCLUDE DATA SCIENCE DISTRIBUTED S
YSTEMS INFORMATION RETRIEVAL COMPUTER NETWORKING INTELLIGENT INFORMATION SYST
EMS AND ALGORITHMS THE DEPARTMENT OFFERS BACHELOR OF SCIENCE MASTER OF SCIENC
E PROFESSIONAL MASTER AND PHD DEGREES PLUS GRADUATE CERTIFICATES ACCELERATED
COURSES AND NONDEGREE STUDY PARTTIME STUDENTS CAN TAKE EVENING CLASSES AND LO
NGDISTANCE STUDENTS CAN EARN MASTERS DEGREES ONLINE STUDENTS RATE OUR TEACHIN
G AS AMONG THE BEST AT THE UNIVERSITY AND OUR FACULTY HAVE WON NUMEROUS TEACH
ING AWARDSTHE SECRET SENTENCE IS GOOD JOB GUYS'
```

Leftover

In [26]:
```
missing_plain = list(alphabets)
for c in cipher_to_plain.items():
    missing_plain.remove(c[1].lower())
missing_plain = [m.upper() for m in missing_plain]

missing_cipher = list(alphabets)
for c in cipher_to_plain.items():
    missing_cipher.remove(c[0])

for m in missing_cipher:
    cipher_to_plain[m] = missing_plain
```

KEY

In [27]:
```
sorted(cipher_to_plain.items())
```

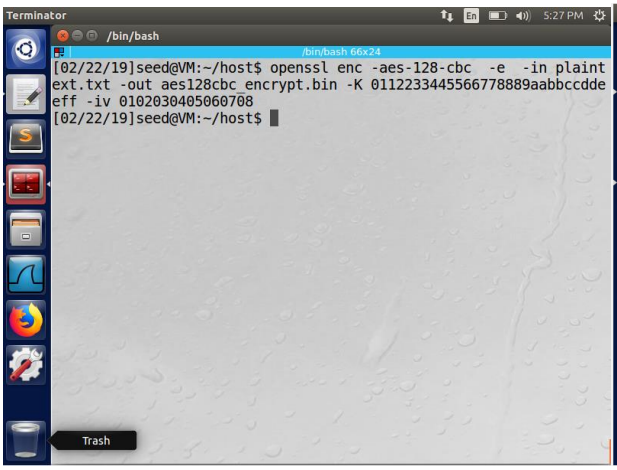
Out[27]:
```
[('a', 'S'),
 ('b', 'J'),
 ('c', 'M'),
 ('d', 'V'),
 ('e', 'B'),
 ('f', 'O'),
 ('g', 'L'),
 ('h', 'C'),
 ('i', 'K'),
 ('j', 'Y'),
 ('k', 'U'),
 ('l', 'N'),
 ('m', 'Q'),
 ('n', 'P'),
 ('o', 'T'),
 ('p', 'E'),
 ('q', 'W'),
 ('r', ['X', 'Z']),
 ('s', 'D'),
 ('t', ['X', 'Z']),
 ('u', 'F'),
 ('v', 'G'),
 ('w', 'R'),
 ('x', 'H'),
 ('y', 'I'),
 ('z', 'A')]
```

2.2 Task 2: Encryption using Different Ciphers and Modes

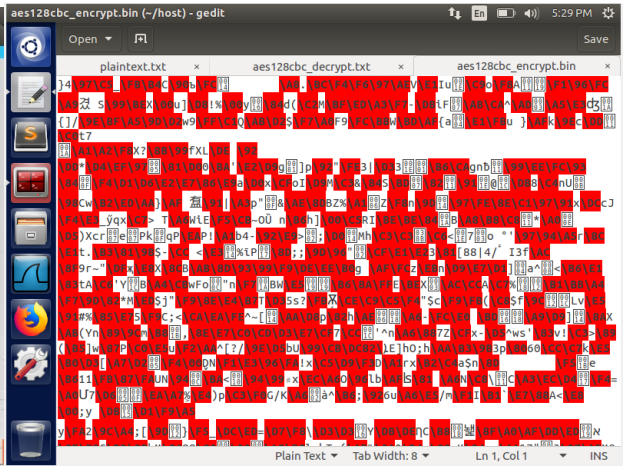
Set plain.txt to plaintext.txt and populated with the result of Task 1

3 Different ciphers and modes:

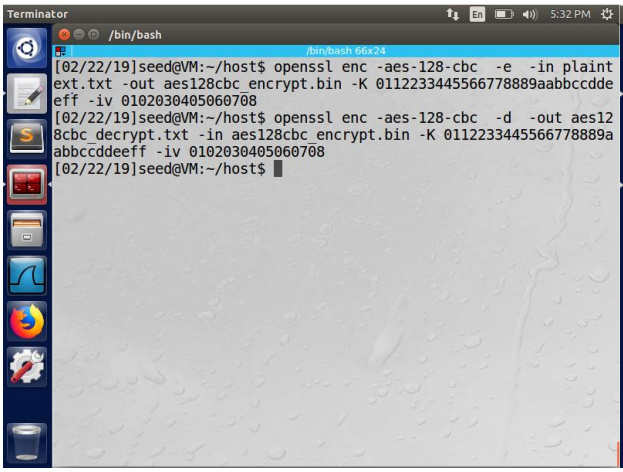
1. AES – 128 – CBC (Cipher: AES, Mode: CBC)



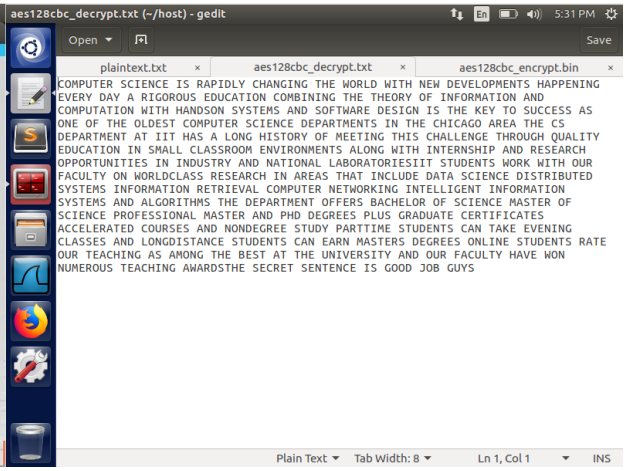
Command



Encryption

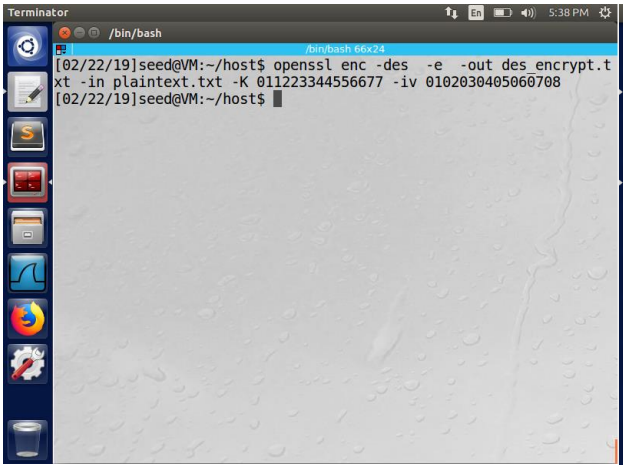


Command

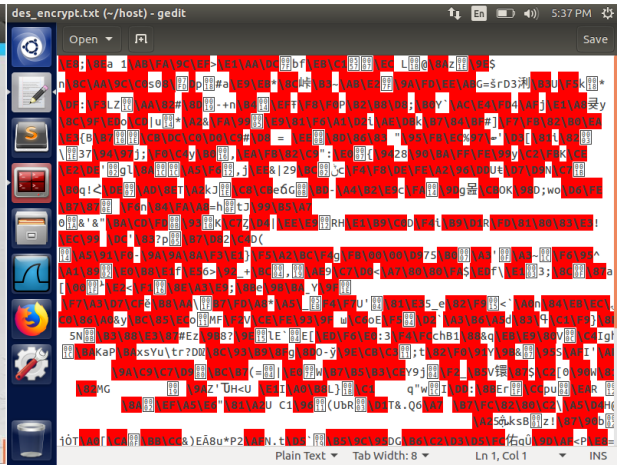


Decryption

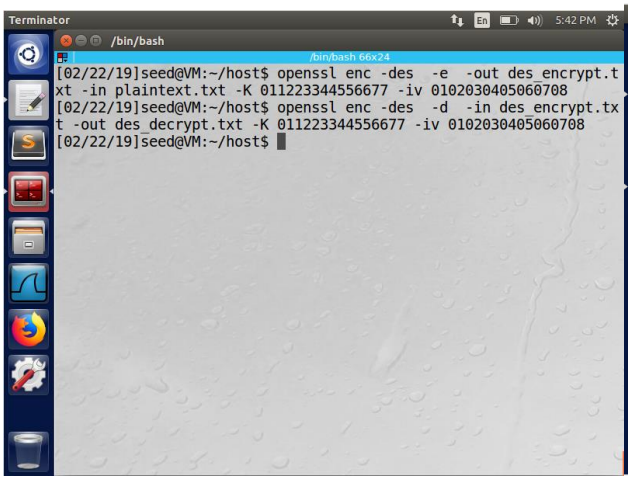
2. DES (Cipher: DES, Mode: ?)



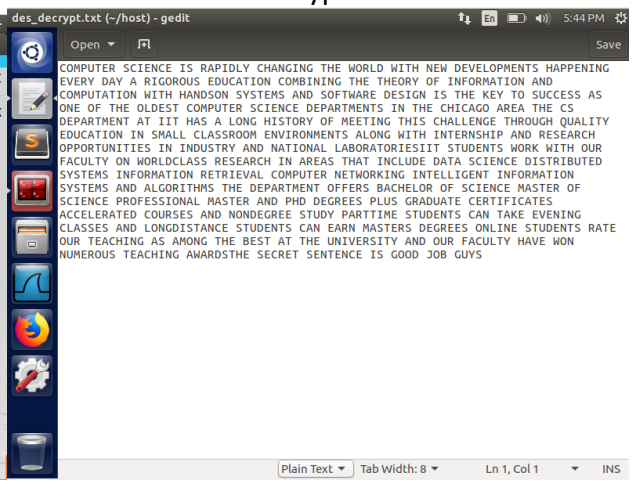
Command



Encryption

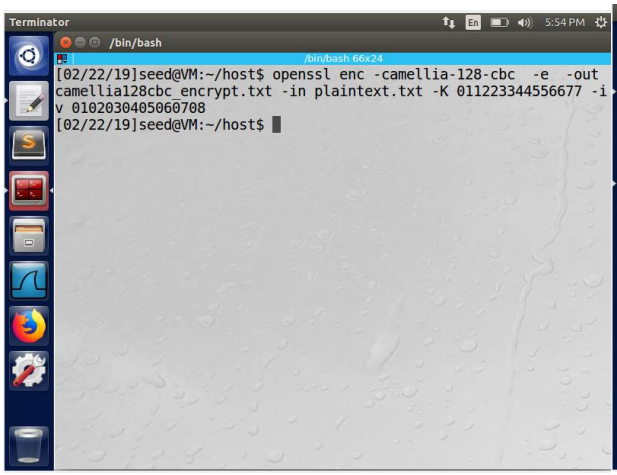


Command

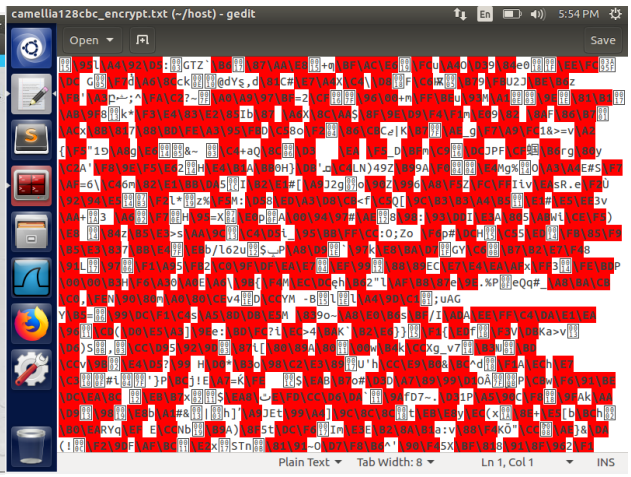


Decryption

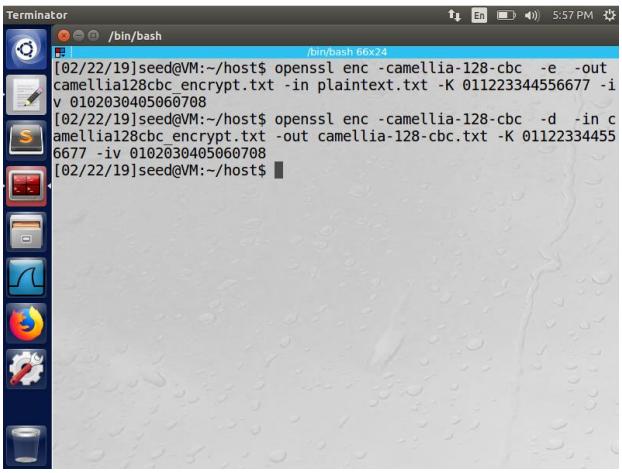
3. Camellia-128-CBC (Cipher: Camellia, Mode: CBC)

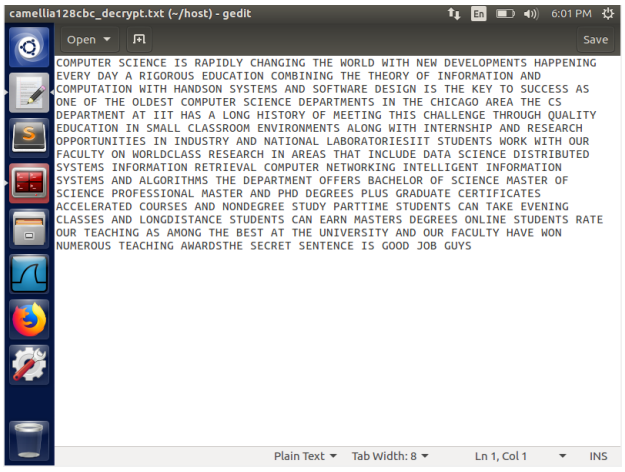


Command



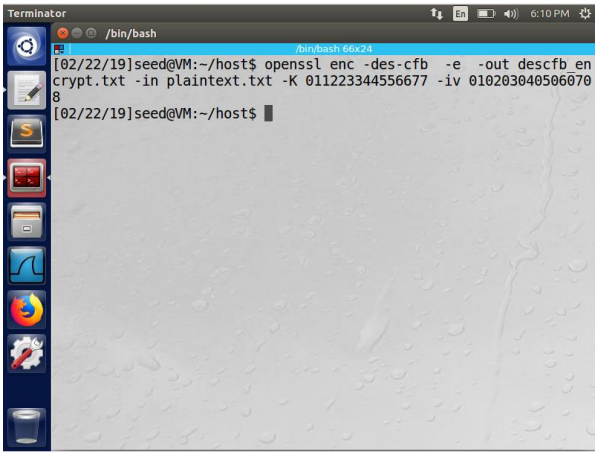
Encryption

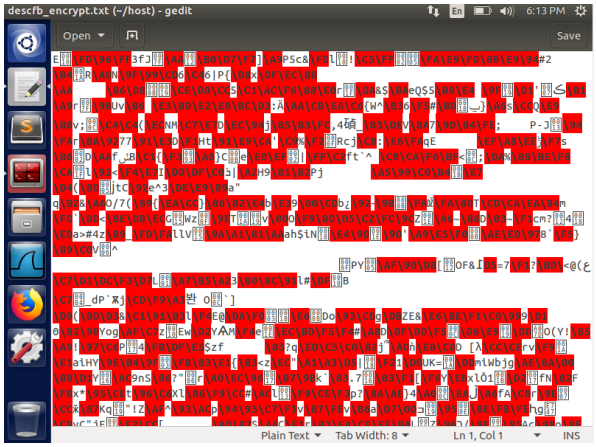




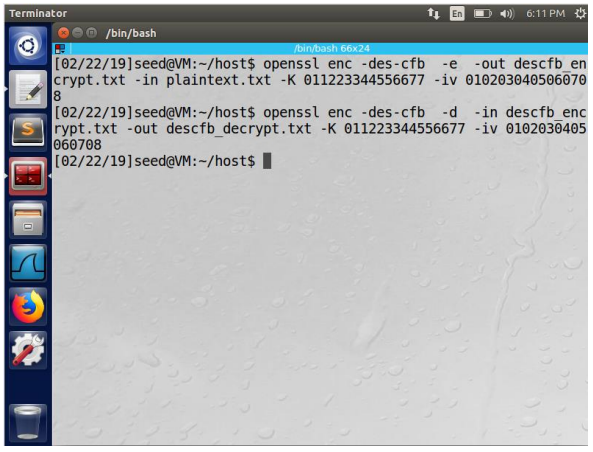
Command Decryption

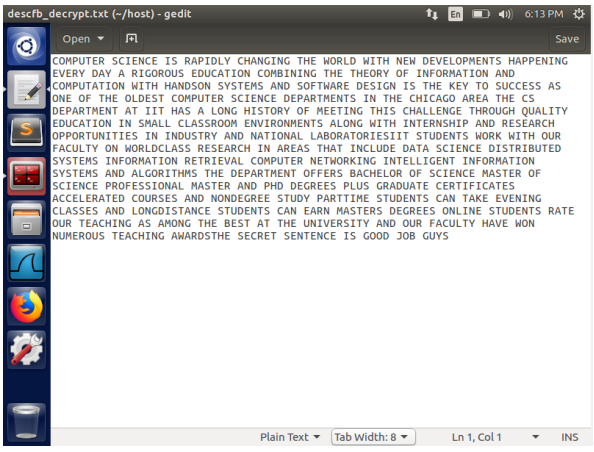
4. DES-CFB (Cipher: DES, Mode: CFB)





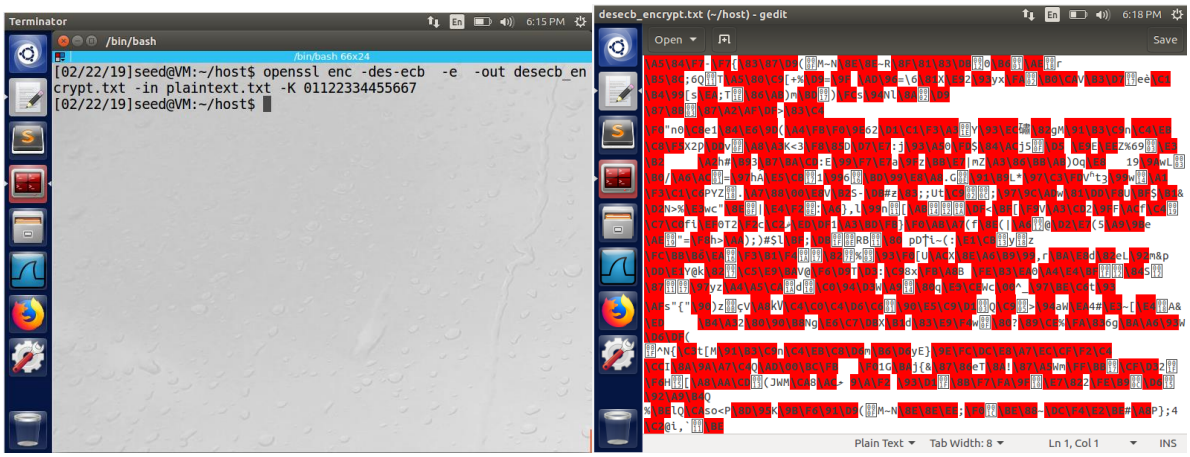
Command Encryption



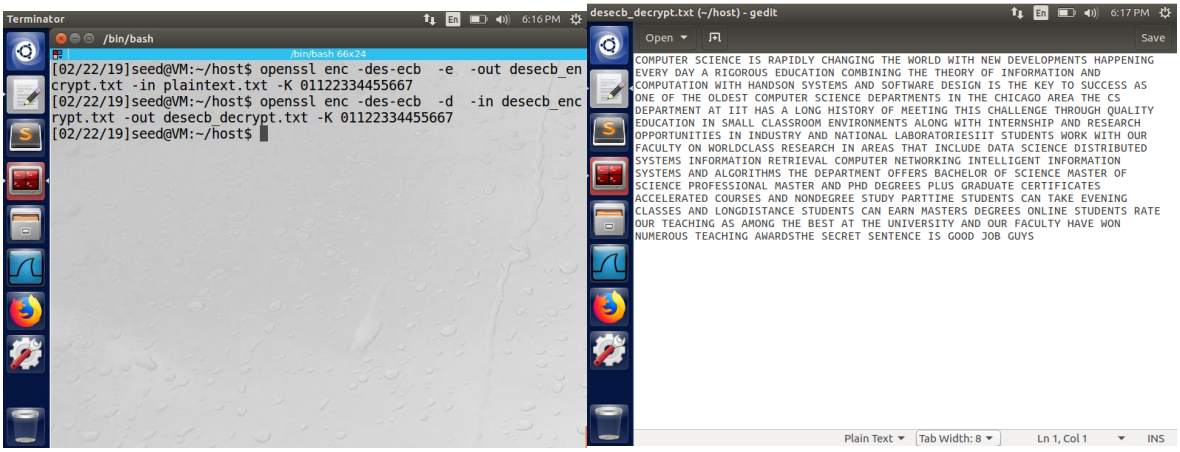


Command Decryption

5. DES-ECB (Cipher: DES, Mode: ECB)



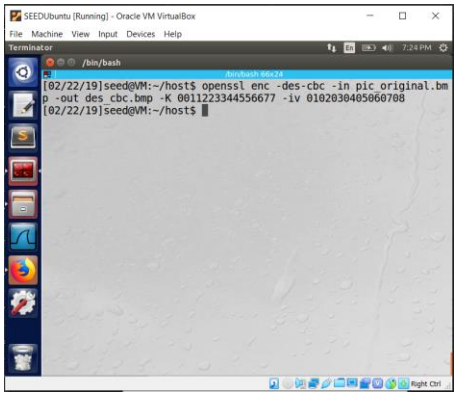
Command Encryption



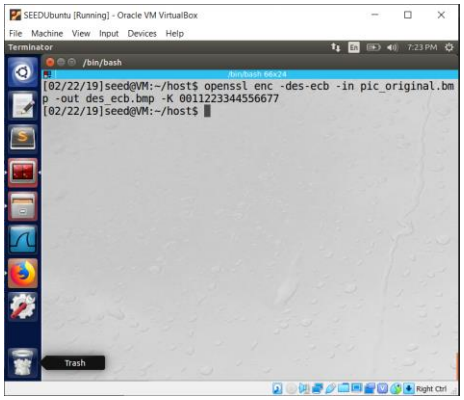
Command Decryption

2.3 Task 3: Encryption Mode – ECB vs. CBC

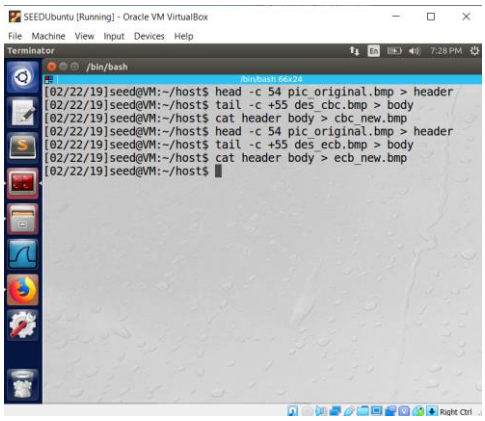
DES-CBC



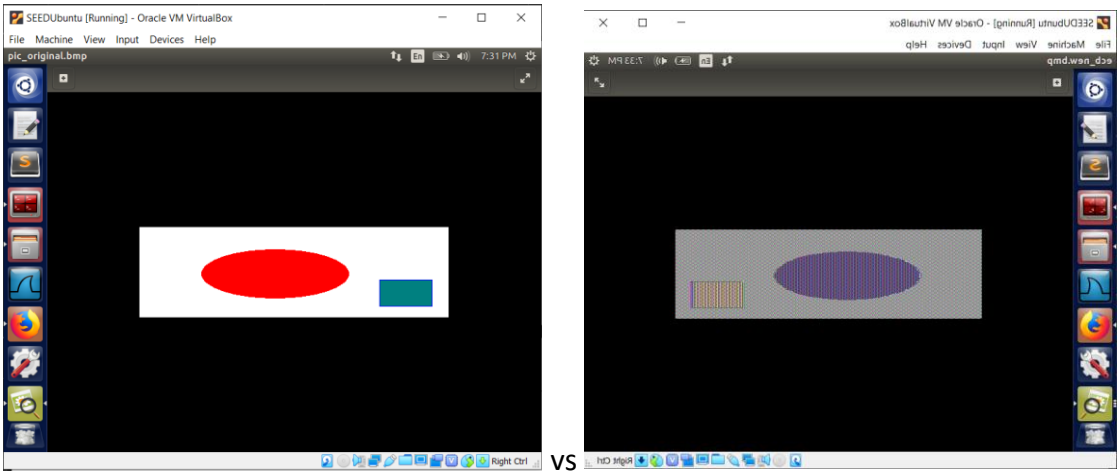
DES-ECB



1. Command

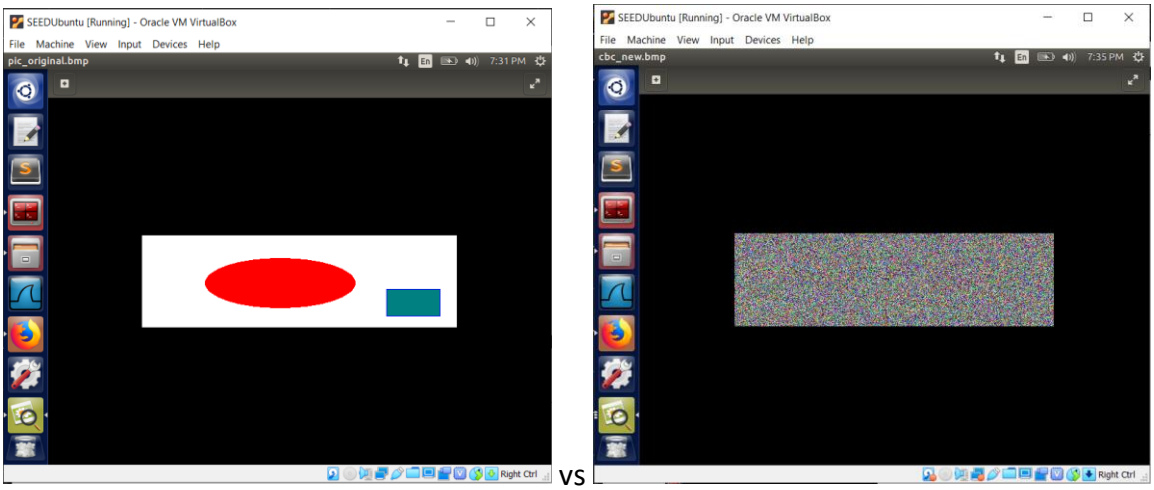


2. Original vs ECB



ECB mode seems to retain the shape of the original image but seems to be mirrored on the y-axis and the colors have been more dimmed.

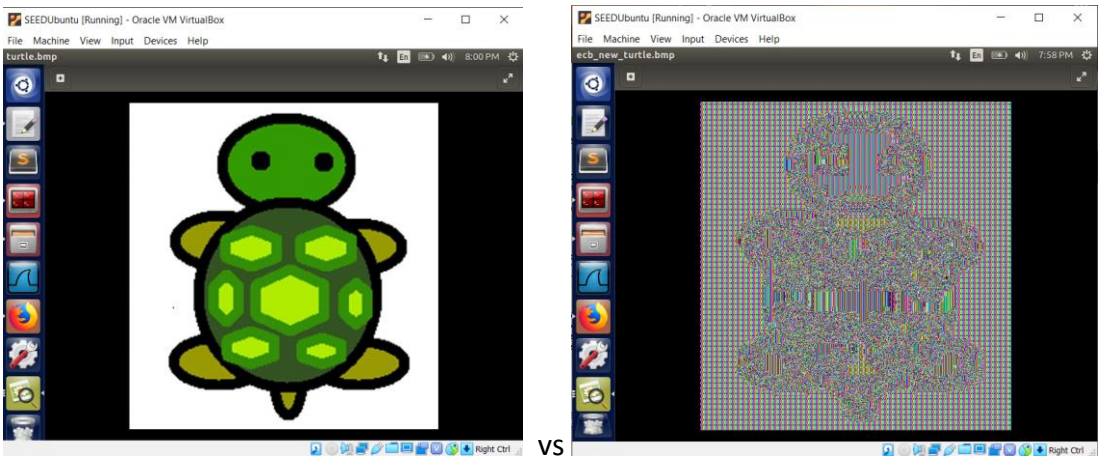
Original vs CBC



CBC has left the image full of noise even though a valid header is given, the original image cannot be retained.

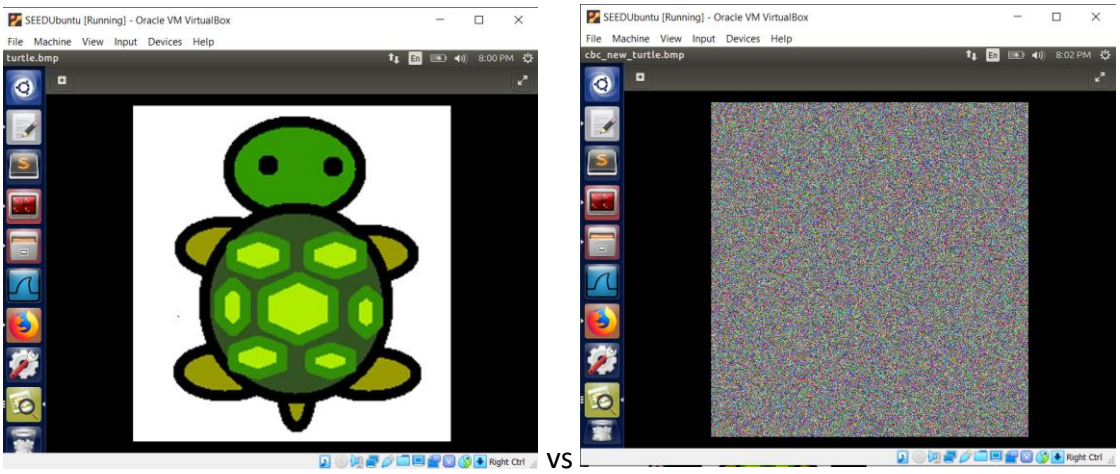
Picture of your choice

Original vs ECB



As before, the mutated ECB header image has retained the shape of the original image but mirroring of the y-axis cannot be evaluated.

Original vs CBC

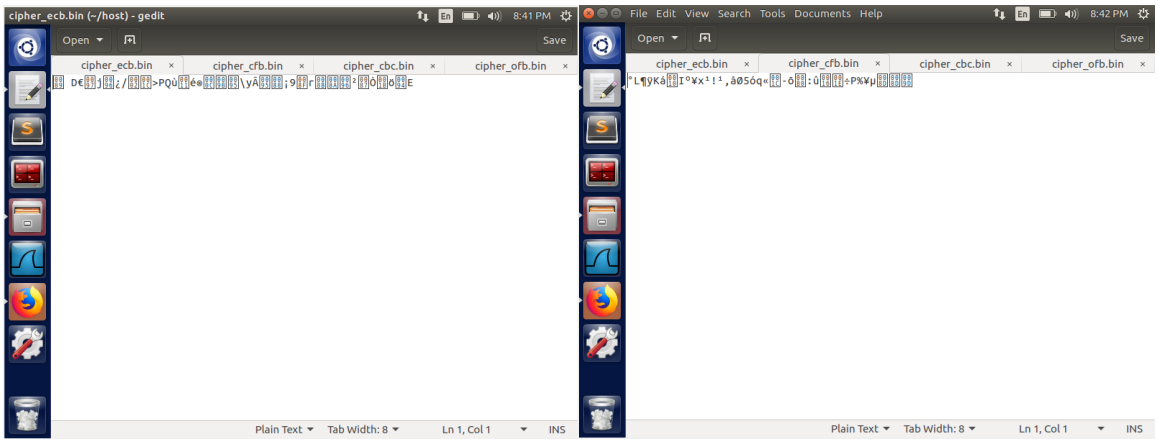


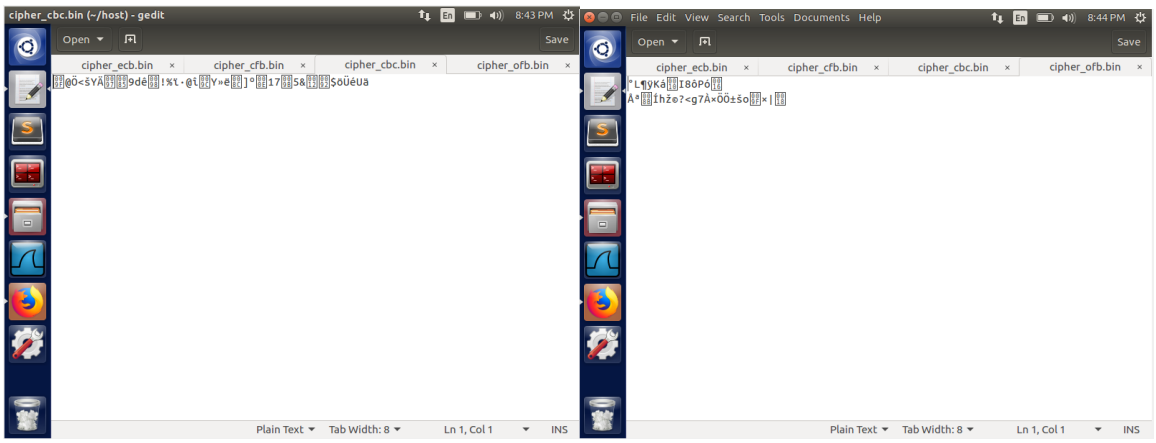
As before, the mutated CBC header image is filled with noise and the shape of the original is not preserved.

2.4 Task 4: Padding

1. Encryptions:

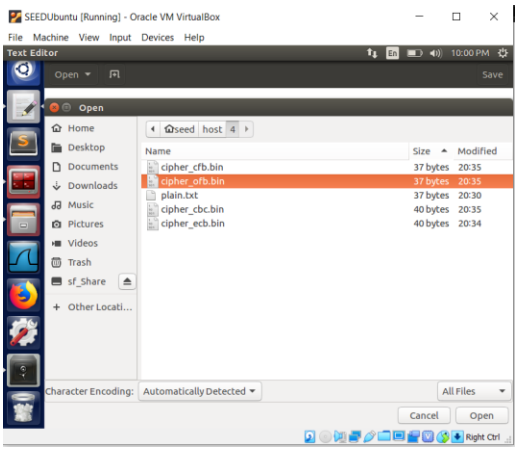
ECB CFB





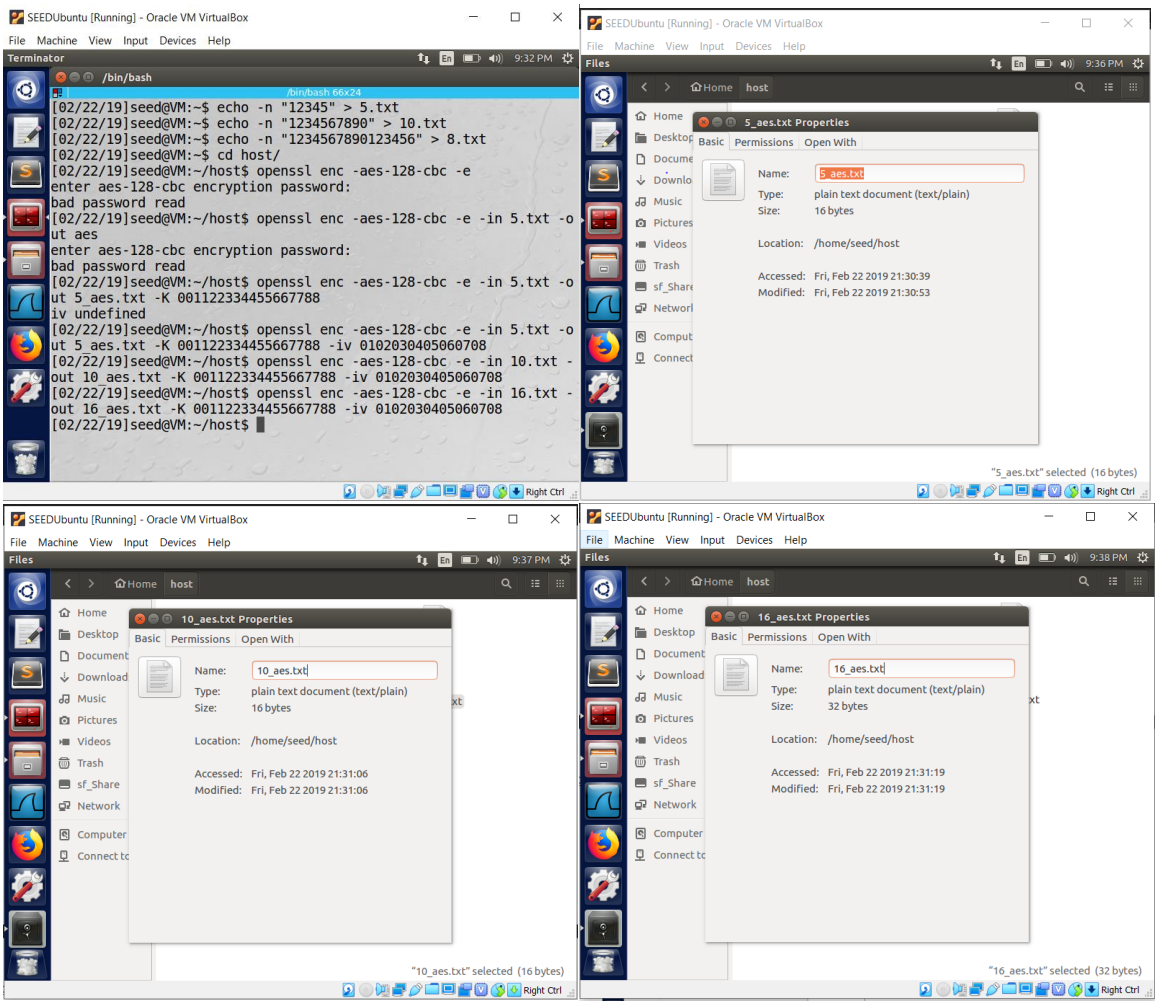
CBC OFB

The CBC and ECB modes has paddings while the CFB and OFB modes do not have paddings since OFB and CFB are using the block cipher as a stream cipher. Since these modes are XORs the plaintext and the block cipher which creates partial blocks.

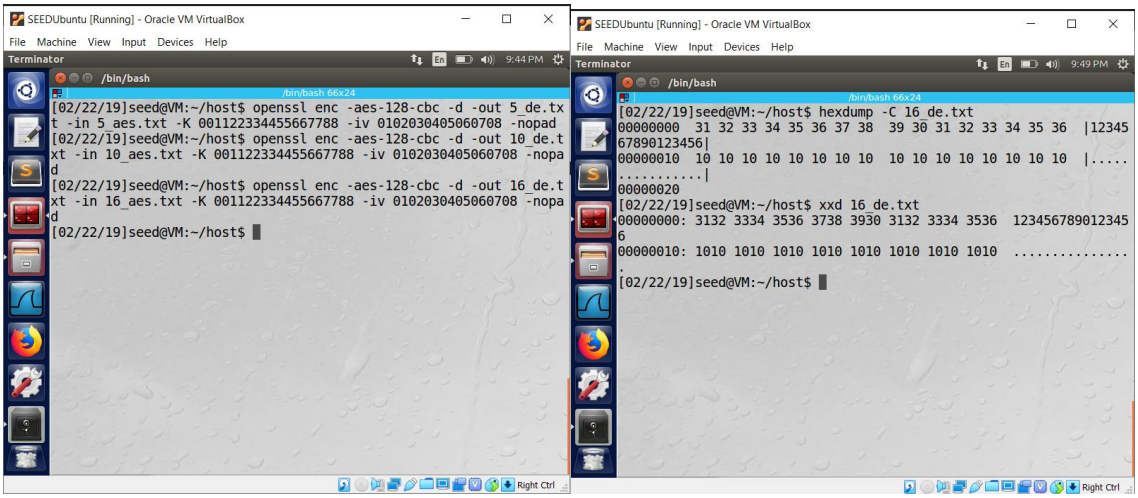


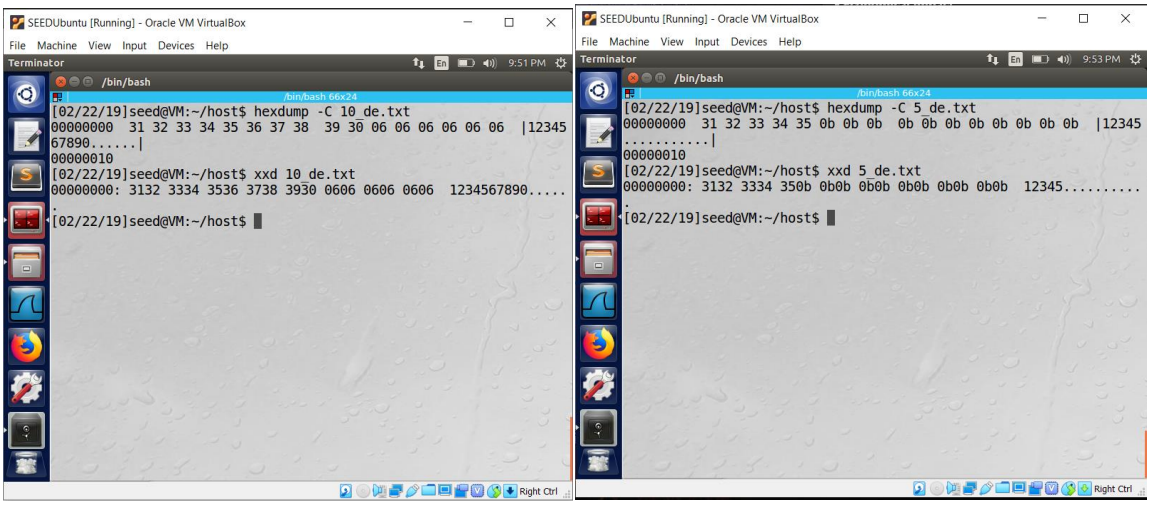
The file sizes also show that CBC and ECB growing in sizes due to padding.

2. Transformations:



Both 5- and 10-byte encrypted files are 16 bytes while 16 byte file became 32 bytes.



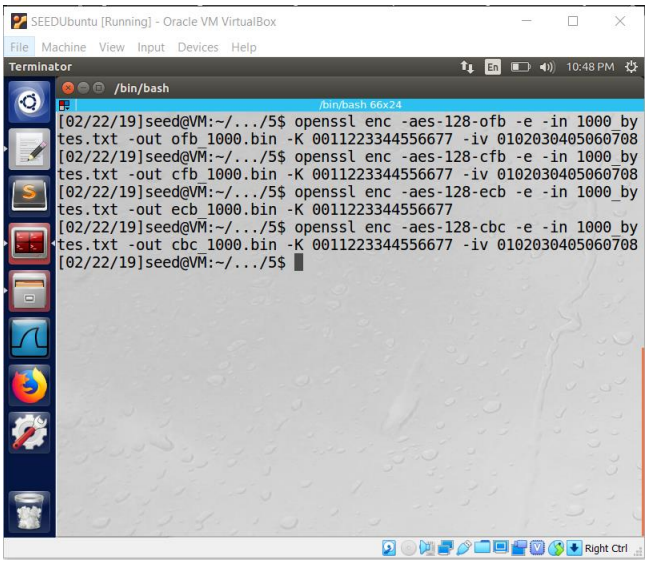


As seen above:

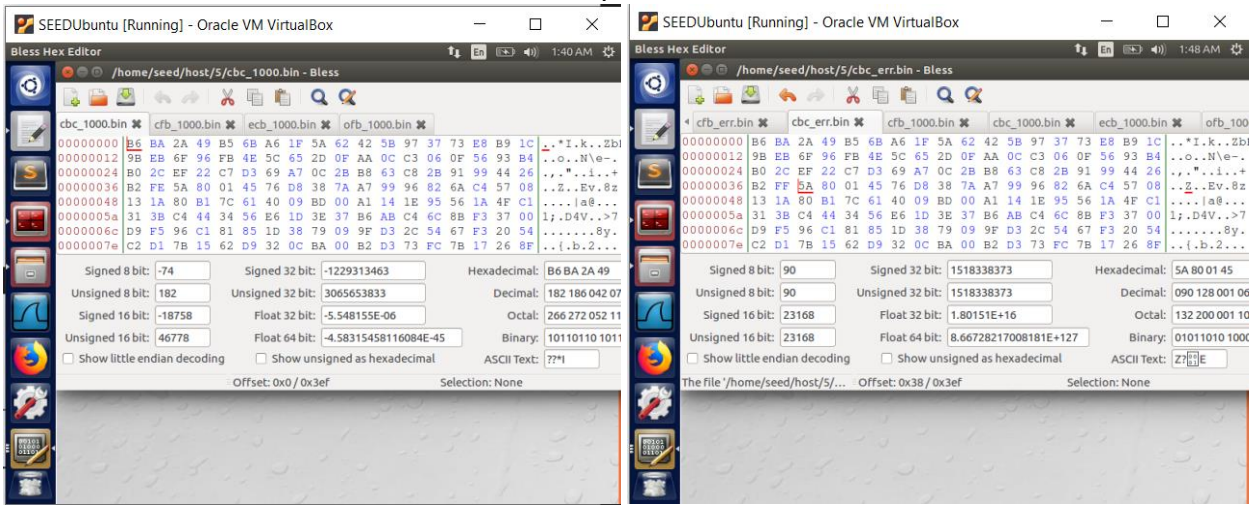
1. 0b (00001011) is added as padding at the end on the byte file 11 times (for 11 bytes = 16 bytes – 6 bytes) [11 = b]
2. 06 (00000110) is added as padding at the end on the byte file 6 times (for 6 bytes = 16 bytes – 10 bytes) [6 = 6]
3. 10 (00010000) is added as padding at the end on the byte file 16 times (for 16 bytes = 32 bytes – 16 bytes) [10 = 16 since 10 doesn't exist in hex (it is a)]

2.5 Task 5: Error Propagation – Corrupted Cipher Text

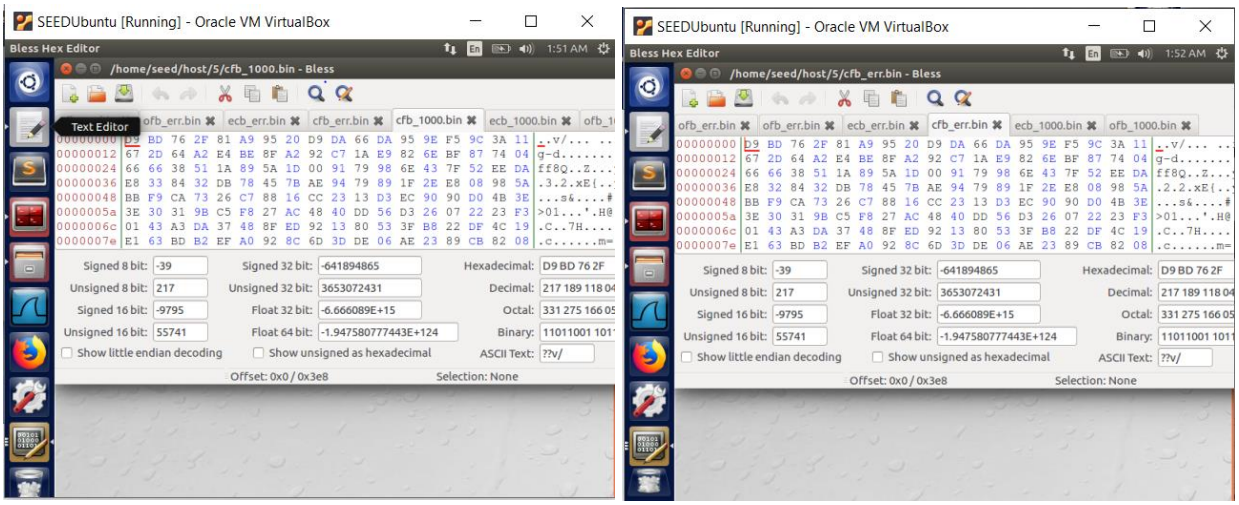
1. ,2.



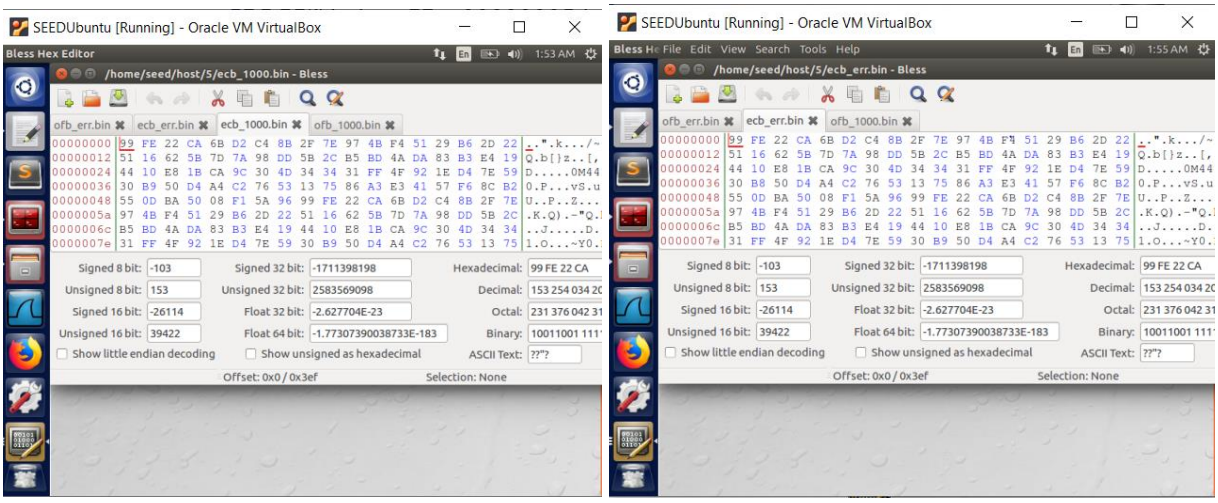
3. 1 bit Substitution in the 55th Byte



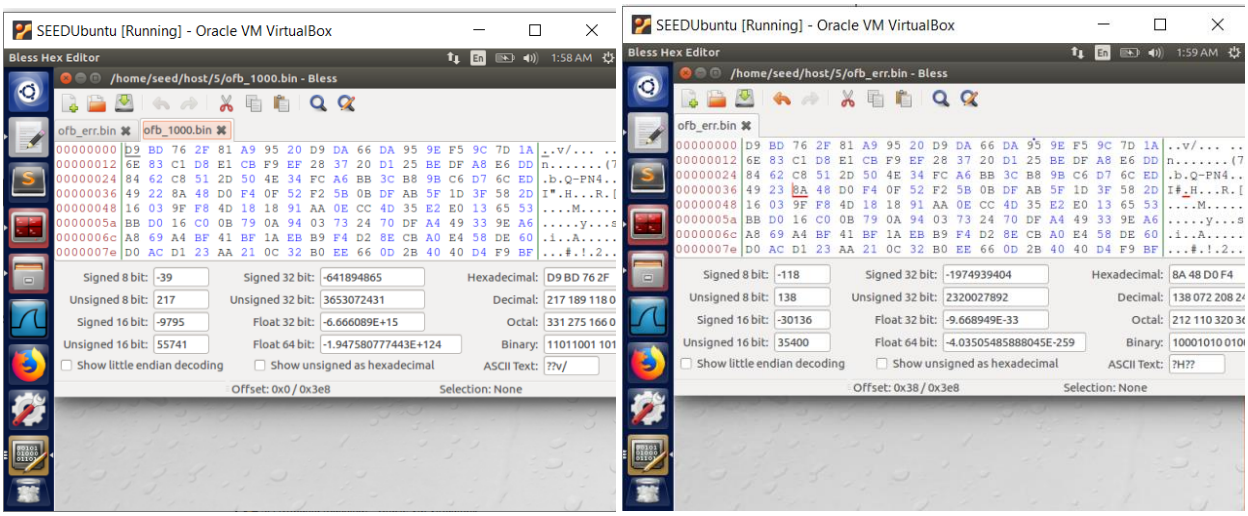
CBC 55th byte = FE = 11111110 to FF = 11111111



CFB 55th byte = 33 = 00110011 to 32 = 00110010

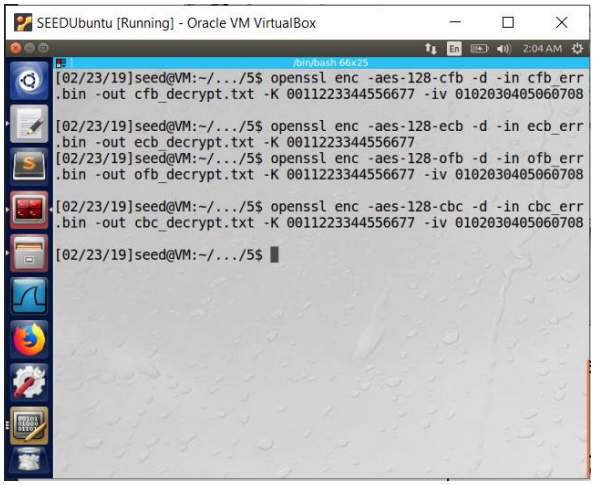


ECB 55th byte = B9 = 10111001 to B8 = 10111000



OFB 55th byte = 22 = 00100010 to 23 = 00100011

4. Decryption

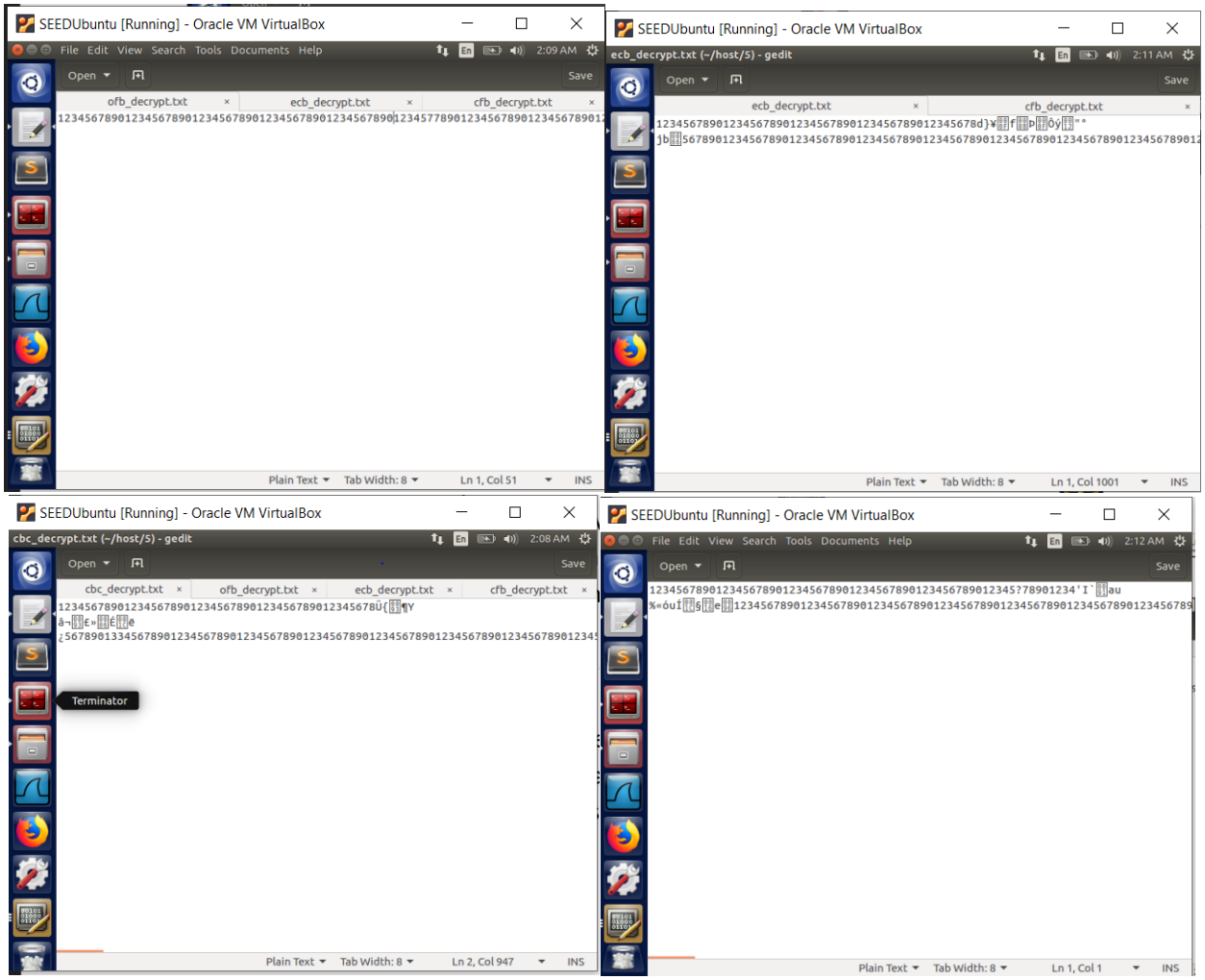


Before looking at the files:

1. CFB should produce the most error since it uses the previous ciphertext as input into the decryption of the next block. Since the error occurs in the 55th bit, the cypher will not self-synchronize until the shift and throw off n/s = 16/4 = 4 blocks of the decryption. (most unrecoverable)
2. CBC should produce the next most errors since the pervious ciphertext is used to decrypt the next block. Thus, the block the error resides and the next block will be error prone (2 blocks).
3. ECB should produce the next least errors since each ciphertext is used only for that block. Thus, the block the error resides should be corrupted (1 block).

4. OFB should produce the least error since each block is independently decrypted without the error propagating the error into incoming blocks; thus, only 1 byte should be impacted. (most recoverable 1 bit)

OFB ECB

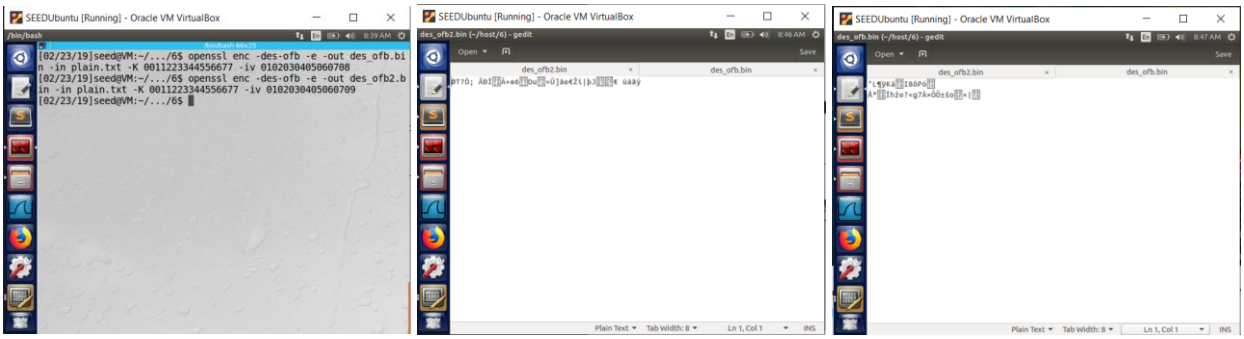


CBC CFB

As justified by previous reasoning, the decrypted text mimics the block corruption. OFB shows 1 bit change, ECB shows 1 block change, CBC shows 2 block change but CFB shows the less corruption which might be due to its synchronization ability.

2.6 Task 6: Initial Vector (IV)

1. (a) Different IVs

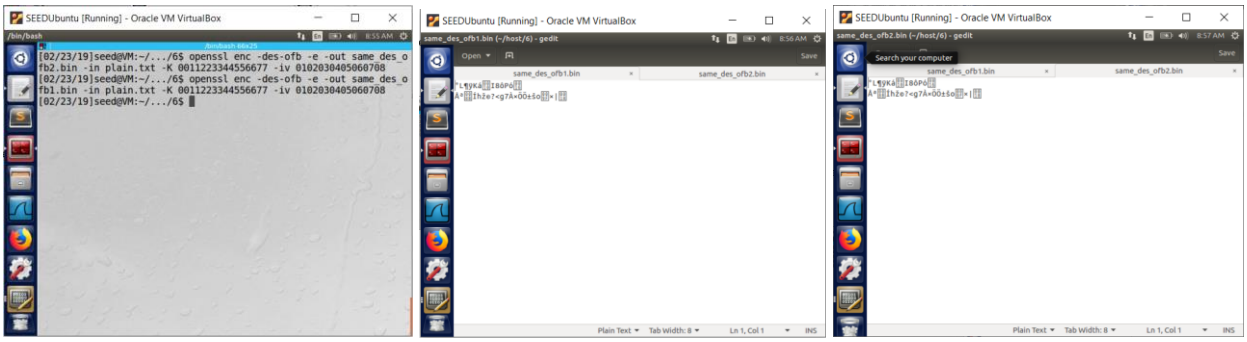


Command Different IV File Different IV File

As observed, encrypted files with different IVs look very different from each other.

(b) Same IVs

1



Command Same IV File Same IV File

As observed, encrypted file with same IVs look the exactly the same as each other.

2. OFB Block:

   Since OFB used encrypted IV and ⊕s the Ciphertext to create the Plaintext in the decryption portion of the OFB mode, we can use this to calculate a message given P_1, C_1 and C_2.

   Thus, the formula is:

   Encrypted IV ⊕ P_1 = C_1

   Encrypted IV = P_1 ⊕ C_1

   Encrypted IV ⊕ P_2 = C_2

   P_2 = Encrypted IV ⊕ C_2

   P_2 = P_1 ⊕ C_1 ⊕ C_2 = 'Order: Launch a missile!\n' = 'Order: Launch a missile!'

CFB Block:

In CFB, XORing the plaintext and ciphertext will result in the key and IV block cipher encryption string; but the same encryption string is not used for every block (the resulting ciphertext is used. Thus, the first block can be easily be recovered but the next blocks will be harder to decipher unless block sizes and encryption algorithm is known.

6.2

In [55]:
```
P1 = 'This is a known message!\n'
C1 = 'a469b1c502c1cab966965e50425438e1bb1b5f9037a4c15913'
```

In [56]:
```
P2 = '?'
C2 = 'bf73bcd3509299d566c35b5d450337e1bb175f903fafc15913'
```

Parse Message

In [57]:
```
import bitarray
```

In [58]:
```
ba = bitarray.bitarray()
ba.frombytes(P1.encode('utf-8'))
P1_binary = list(ba)
```

In [59]:
```
C1_binary = [True if b == '1' else False for b in bin(int(C1, 16))[2:]]
```

IV

In [60]:
```
IV = [C1_binary[c] != P1_binary[c] for c in range(len(C1_binary))]
```

Translation

In [61]:
```
C2_binary = [True if b == '1' else False for b in bin(int(C2, 16))[2:]]
```

In [62]:
```
P2_binary = [C2_binary[c] != IV[c] for c in range(len(C2_binary))]
P2_binary_str = ''
for p in P2_binary:
    if p == True:
        P2_binary_str += '1'
    else:
        P2_binary_str += '0'
```

In [63]:
```
P2_hex = hex(int(P2_binary_str, 2))[2:]
```

In [64]:
```
P2 = bytearray.fromhex(P2_hex).decode()
P2
```

Out[64]:
```
'Order: Launch a missile!\n'
```

3. Predictable IVS:

   Key = '00112233445566778899aabbccddeeff'

   IV_1 = '31323334353637383930313233343536'

   IV_2 = '31323334353637383930313233343537'

   C_1 = ‘bef65565572ccee2a9f9553154ed9498’

   Following CBC block mode, we can use the following formula:

   $P_1 \oplus IV_1 = P_2 \oplus IV_2 = X$ such that X can sent to the AES-128 algorithm and $C_1 = C_2$

   Since we know that P1 can be ‘Yes’ or ‘No’, let us assume P1 = ‘Yes’:

   P_1 = ‘Yes’

   P_1_hex = ‘5965730d0d0d0d0d0d0d0d0d0d0d0d0d’

   Since the blocks are 16 bytes long and Yes makes up 3 bytes, 16 - 3 = 13 = 0d will be the padding add 13 times.

   Now solve for P2:

   $P_1 \oplus IV_1 = X = P_2 \oplus IV_2$

   0x68574039383b3a35343d3c3f3e39383b = $X = P_2 \oplus IV_2$

   $P_2 = X \oplus IV_2$

   P_2 = 0x5965730d0d0d0d0d0d0d0d0d0d0d0d0c

   P_2 = ‘Yes\r\r\r\r\r\r\r\r\r\r\r\x0c’

   We can ask Bob to encrypt P_2 = ‘Yes\r\r\r\r\r\r\r\r\r\r\r\x0c’. This should yield in a C_2 and we can compare with C_1. If $C_1 = C_2$, then P_1 = ‘Yes’, otherwise P_1 = ‘No’. Using bliss, to edit a file to enter P_2, we get the following:

6.3

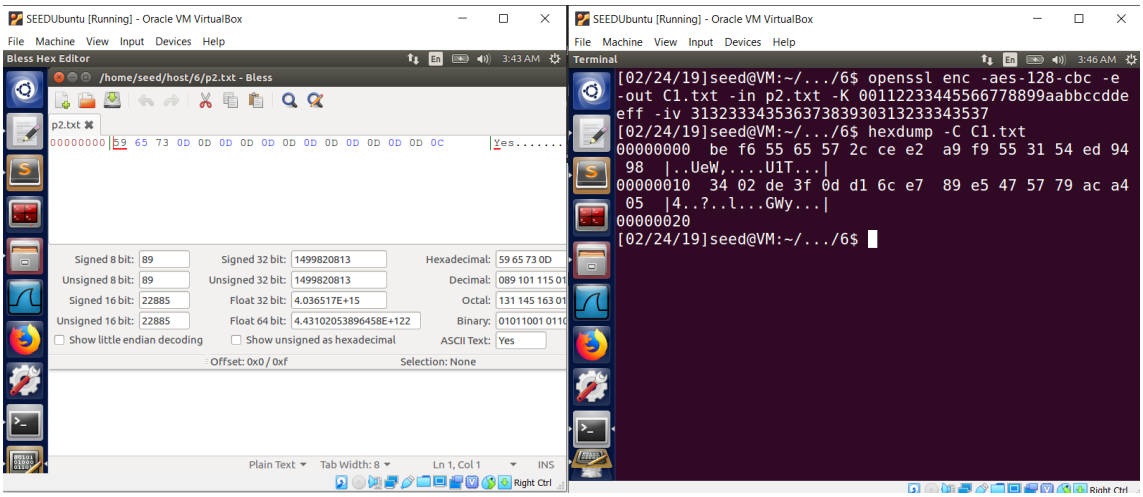
In [66]:
```
import bitarray
P1 = 'Yes'
oIV = '31323334353637383930313233343536'
nIV = '31323334353637383930313233343537'
P2 = '?'
```

In [76]:
```
ba = bitarray.bitarray()
ba.frombytes(P1.encode('utf-8'))
P1_binary = list(ba)

oIV_binary = [True if b == '1' else False for b in bin(int(oIV, 16))]
nIV_binary = [True if b == '1' else False for b in bin(int(nIV, 16))]


for f in range(int((len(oIV_binary) - len(P1_binary))/8)):
    P1_binary += [0,0,0,0,1,1,0,1]

X = [oIV_binary[c] != P1_binary[c] for c in range(len(oIV_binary))]
X_str = ''
for x in X:
    if x == True:
        X_str += '1'
    else:
        X_str += '0'
X_hex = hex(int(X_str, 2))[2:]
print("X HEX: ",X_hex)

P2_binary = [X[c] != nIV_binary[c] for c in range(len(nIV_binary))]

P2_binary_str = ''
for p in P2_binary:
    if p == True:
        P2_binary_str += '1'
    else:
        P2_binary_str += '0'

P2_hex = hex(int(P2_binary_str, 2))[2:]
print("P2 HEX: ",P2_hex)
P2 = bytearray.fromhex(P2_hex).decode()
P2
```

```
X HEX:  68574039383b3a35343d3c3f3e39383b
P2 HEX:  5965730d0d0d0d0d0d0d0d0d0d0d0d0c
```

Out[76]:
```
'Yes\r\r\r\r\r\r\r\r\r\r\r\r\x0c'
```

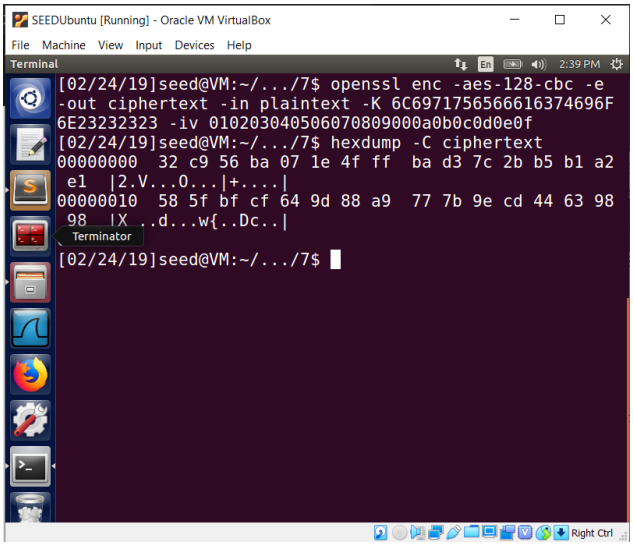
In [ ]:
```
```



As seen, the hex dump is: be f6 55 65 57 2c ce e2 a9 f9 55 31 54 ed 94 98 for the first block. This is the same as C_1 and thus results in P_1 = 'Yes'.

2.7 Task 7: Programming using the Crypto Library (EXTRA CREDIT):

1. Set the message in char []
2. Set IV as hex in char [] using 0x..
3. Set key in char [] and transform to char [] filled with integer of each character
4. Pass all arguments to EVP commands
5. Convert resulting ciphertext to char [] filled with hex without 0x
6. Compare resulting hex [] with the given ciphertext.
7. Code is more commented
8. Resulting KEY: **liquefaction**
9. Test KEY -> liquefaction = liquefaction#### = **6c697175656661637469 6f6e23232323**



10. Resulting Ciphertext = 32c956ba07le4fffbad37c2bb5b1a2e1
585fbfcf649d88a9777b9ecd44639898 (WHICH IS A MATCH)

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <openssl/evp.h>

int main()
        {
        int i,j;
    FILE *fp;
    int outlen, tmplen, inlen;

    unsigned char outbuf[65];                   // RESULTING CIPHERTEXT
    unsigned char outbuf_hex[65];               // CONVERT ASCII CIPHERTEXT TO HEX CIPHERTEXT
    unsigned char key[16];                              // KEY INITIALIZE
    unsigned char key_int[16];                  // INT VERSION OF CHAR [] OF KEY

    EVP_CIPHER_CTX ctx;
    unsigned char iv[] = {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x00,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f};
    unsigned char C[] = "32c956ba071e4fffbad37c2bb5b1a2e1585fbfcf649d88a9777b9ecd44639898";
    char intext[22] = "This is not a secret.";

    fp = fopen("words.txt", "r");               // READ WORDS TEXT FILE

    if (fp == NULL)                                             // CHECK FILE
    {
        perror("Error while opening the file.\n");
        exit(EXIT_FAILURE);
    }

    while(fgets(key, sizeof key, fp)){  // LOOP THROUGH KEYS

                memset(outbuf_hex,0,sizeof(outbuf_hex));        // EMPTY OUT ARRAYS...
                memset(outbuf,0,sizeof(outbuf));
                memset(key_int,0,sizeof(key_int));

                key[strlen(key) - 1] = '#';                                     // CONVERT \N TO #

                for(i = 0; i < strlen(key); i++){                       // LOOP THROUGH KEY CHAR []
                        key_int[i] = (int)key[i];                               // CONVERT ALL CHAR KEY TO INT
                }

                for(i = strlen(key); i < 16; i++){                      // LOOP TO REMAINING KEY LENGHT (REACH 128)
                        key_int[i] = 35;                                                // APPEND # -> 0X23 -> 35 AS INT
                }

                EVP_CIPHER_CTX_init(&ctx);                                      // INITIALIZE EVP CTX
                EVP_EncryptInit_ex(&ctx, EVP_aes_128_cbc(), NULL, key_int, iv);         //PASS ARGUEMENTS

                if(!EVP_CipherUpdate(&ctx, outbuf, &outlen, intext, strlen(intext))) //UPDATE OUTBUF WITH CIPHERTEXT
                {
                        /* Error */
                        printf("ERROR!");
                        return 0;
                }

        if(!EVP_CipherFinal_ex(&ctx, outbuf + outlen, &tmplen))         // FINALIZE CIPHERTEXT
                {
                /* Error */
                return 0;
        }
        EVP_CIPHER_CTX_cleanup(&ctx);                           // CLEAR OUT EVP.CTX

                for(i = 0, j = 0; i < strlen(outbuf); i++,j+=2){        // LOOP THROUGH CIPHERTEXT
                        sprintf((char*)outbuf_hex+j,"%02x",outbuf[i]);  // CONVERT ASCII TO HEX W/O 0X..
                }

                if(!(strcmp(outbuf_hex,C))){            // CHECK IF CIPHERTEXTS MATCH
                        key[strlen(key) - 1] = 0;
                        printf("KEY: %s\n",key);                // PRINT KEY
                        break;                                                  // EXIT OUT OF LOOP
                }

    }
    fclose(fp);                                                         // CLOSE WORDS.TXT
    return 0;
        }
```