Activity IV - Fundamental of Cryptography

Created by : Krerk Piromsopa, Ph.D

Overviews

In this activity, we will learn the basics of encryption. There are 3 exercises in this activity. Each exercise is designed to let you learn the concepts of cryptography.

We will need:

- Imagemagick
- OpenSSL
- One of your favorite programming languages.

You are welcome to do this exercise with any programming language. If you have no preference, use python.

Exercises

1. (Encryption and Statistical Analysis) Though encryption is primarily designed to preserve confidentiality and integrity of data, the mechanism itself is vulnerable to brute force (statistical analysis). In other words, the more we see the encrypted data, the easier we can hack it. In this exercise, you are asked to crack the following cipher text. Please provide the decrypted result and explain your strategy in decrypted this text.

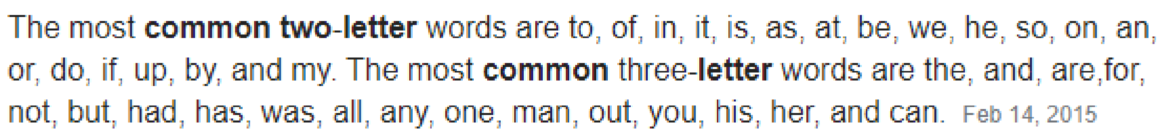
Cipher text

PRCSOFQX FP QDR AFOPQ CZSPR LA JFPALOQSKR. QDFP FP ZK LIU BROJZK MOLTROE.

a. Count the frequency of letters. List the top three most frequent characters.
top three are P(7) F,O,R(6) Q(5)

```
A has 3 occurence
B has 1 occurence
C has 2 occurence
D has 2 occurence
E has 1 occurence
F has 6 occurence
I has 1 occurence
J has 2 occurence
K has 3 occurence
L has 4 occurence
M has 1 occurence
O has 6 occurence
P has 7 occurence
Q has 5 occurence
R has 6 occurence
S has 3 occurence
T has 1 occurence
U has 1 occurence
X has 1 occurence
Z has 3 occurence
```

b. Knowing that this is English, what are commonly used three-letter words and two-letter words. Does the knowledge give you a hint on cracking the given text.

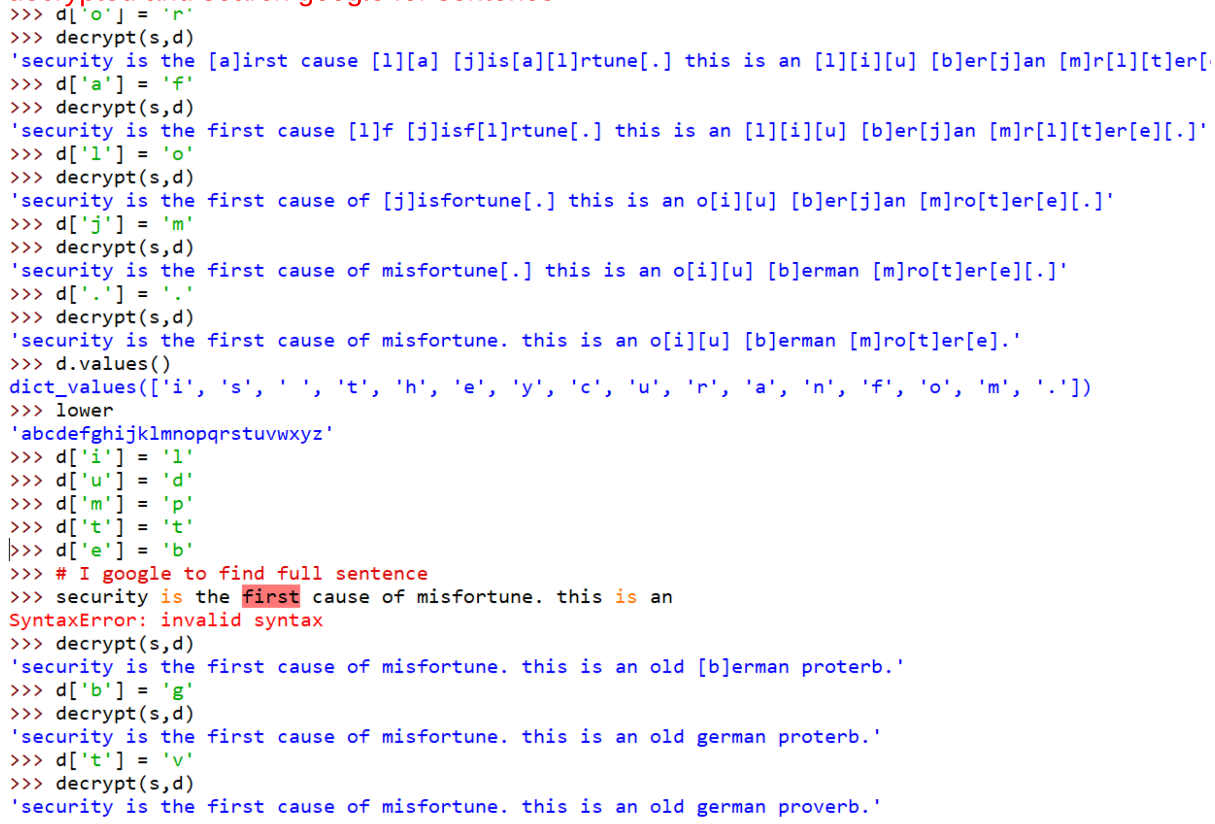

The most **common two-letter** words are to, of, in, it, is, as, at, be, we, he, so, on, an, or, do, if, up, by, and my. The most **common** three-**letter** words are the, and, are,for, not, but, had, has, was, all, any, one, man, out, you, his, her, and can. Feb 14, 2015

yes help a lot

c. Cracking the given text. Measure the time that you have taken to crack this message.
'security is the first cause of misfortune. this is an old german proverb.'
9.14 minutes

d. Explain your process in hacking such messages.
I write script that decrypt string with input dict indicate translate pair and try guessing pair. Up to certain point I can't guess so I take the part that is

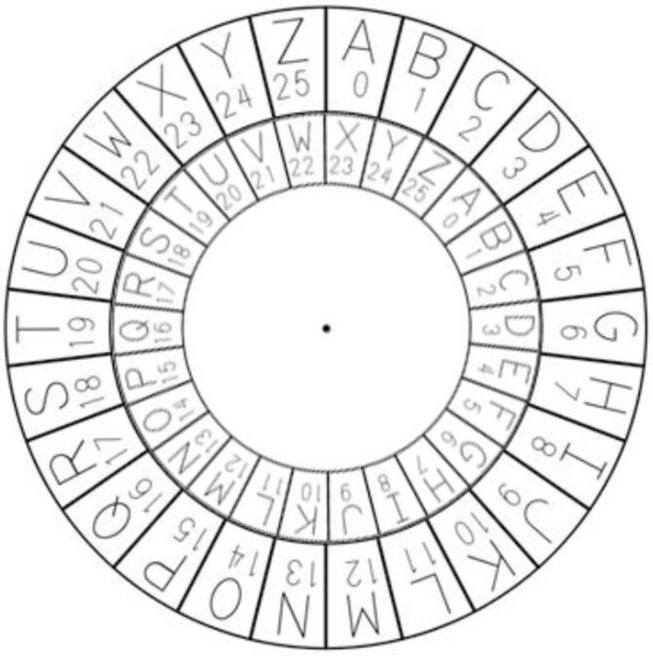
decrypted and search google for sentence



```
>>> d['o'] = 'r'
>>> decrypt(s,d)
'security is the [a]irst cause [l][a] [j]is[a][l]rtune[.] this is an [l][i][u] [b]er[j]an [m]r[l][t]er[
>>> d['a'] = 'f'
>>> decrypt(s,d)
'security is the first cause [l]f [j]isf[l]rtune[.] this is an [l][i][u] [b]er[j]an [m]r[l][t]er[e][.]'
>>> d['l'] = 'o'
>>> decrypt(s,d)
'security is the first cause of [j]isfortune[.] this is an o[i][u] [b]er[j]an [m]ro[t]er[e][.]'
>>> d['j'] = 'm'
>>> decrypt(s,d)
'security is the first cause of misfortune[.] this is an o[i][u] [b]erman [m]ro[t]er[e][.]'
>>> d['.'] = '.'
>>> decrypt(s,d)
'security is the first cause of misfortune. this is an o[i][u] [b]erman [m]ro[t]er[e].'
>>> d.values()
dict_values(['i', 's', ' ', 't', 'h', 'e', 'y', 'c', 'u', 'r', 'a', 'n', 'f', 'o', 'm', '.'])
>>> lower
'abcdefghijklmnopqrstuvwxyz'
>>> d['i'] = 'l'
>>> d['u'] = 'd'
>>> d['m'] = 'p'
>>> d['t'] = 't'
>>> d['e'] = 'b'
>>> # I google to find full sentence
>>> security is the first cause of misfortune. this is an
SyntaxError: invalid syntax
>>> decrypt(s,d)
'security is the first cause of misfortune. this is an old [b]erman proterb.'
>>> d['b'] = 'g'
>>> decrypt(s,d)
'security is the first cause of misfortune. this is an old german proterb.'
>>> d['t'] = 'v'
>>> decrypt(s,d)
'security is the first cause of misfortune. this is an old german proverb.'
```

e. If you know that the encryption scheme is based on Caesar Wheel (Monoalphabetic Substitution) that is commonly used by Caesar for sending messages to Cicero, does it allow you to crack it faster?
yes because I only have to align wheel that can translate common word such as ‘the’ ‘is’

f. Draw a cipher disc of the given text.
May I use PhotoShop if you don't mind



g. Create a simple python program for cracking the Caesar Wheel cipher text using brute force attack. Explain the design and demonstrate your software. (You may use an English dictionary for validating results.)

```python
import enchant
import string
d = enchant.Dict("en_US")

cipher = input().tolower()

maxvalid = 0
ans = ''
for i in range(26):

    plain = ''
    for a in cipher:
        if a in string.ascii_lowercase:
            plain += chr((ord(a)+i-97)%26 + 97)
        else:
            plain += a
    valid = sum([d.check(word) for word in plain.split().strip('.
')])
    if valid > maxvalid:
        ans = plain
        maxvalid = valid
print(ans)
```

2. (Symmetric Encryption) Vigenère is a complex version of Caesar Wheel cipher. It is a polyalphabetic substitution.

a. Based on the Confederate Cipher Disc, explain how it can be used to cipher data.
The mapping is depend on what is the alphabet of the key that align with considering alphabet in cipher text. To decrypt is to align ‘a’ letter with current key alphabet and map current letter in cipher back to plain according to the disc.

b. If a key is the word “CAT”, how many cipher discs do we need? Please analyze the level of security provided by Vigenère compared to that of the Caesar Wheel.
actually one and spinning around every letter or three if you don’t want to spin every letter.
Compare to Caesar where only one spin can solve the cipher data.

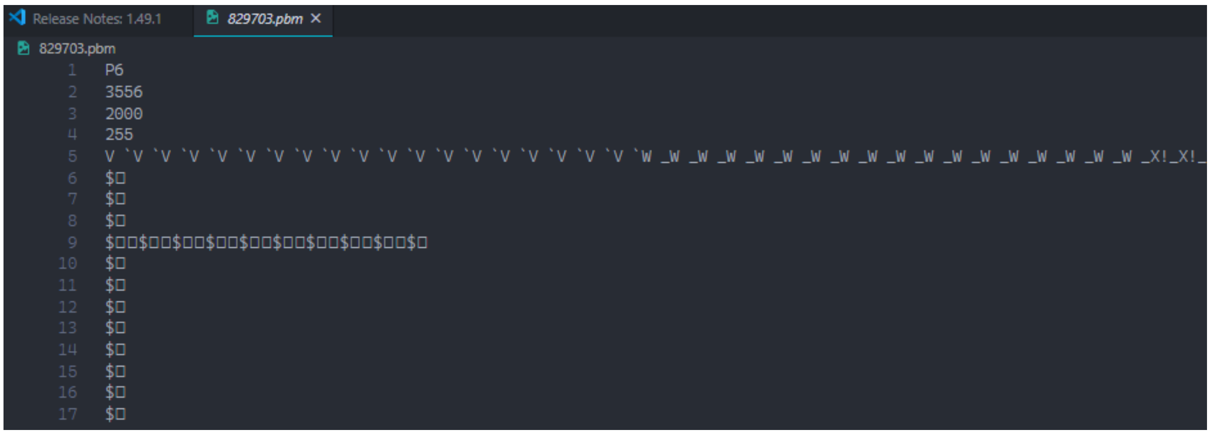
c. Create a python program for ciphering data using Vigenère.

```python
import string
key = input.strip().lower()
plain = input.strip().lower()

cipher = ''
for (i,a) in enumerate(plain):
    if a in string.ascii_lowercase:
        currentkey = key[i%len(key)]
        offset = currentkey - 97
        cipher += chr(((ord(a) + offset)- 97)%26 + 97)
    else:
        cipher += a

print(cipher)
```

17. (Mode in Block Cipher) Block Cipher is designed to have more randomness in a block. However, an individual block still utilizes the same key. Thus, it is recommended to use a cipher mode with an initial vector, chaining or feedback between blocks. This exercise will show you the weakness of **E** lectronic **C** ode **B** ook mode which does not include any initial vector, chaining or feedback.

    a. Find a bitmap image that is larger than 2000x2000 pixels. Note that you may resize any image. To simplify the pattern, we will change it to bitmap (1-bit per pixel) using the portable bitmap format (pbm). In this example, we will use imagemagick for the conversion.

    ```
    $ convert image.jpg -resize 2000x2000 org.pbm
    ```

    ```
    P6
    3556
    2000
    255
    V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _X!_X!_
    $□
    $□
    $□
    $□□$□□$□□$□□$□□$□□$□□$□□$□□$□
    $□
    $□
    $□
    $□
    $□
    $□
    $□
    $□
    ```

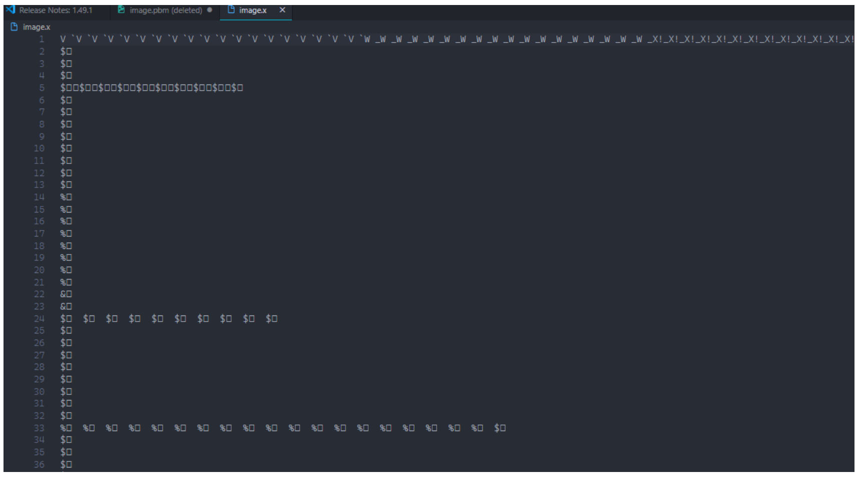
    b. The NetPBM[1] format is a naive image format.The first two lines contain a header (format and size in pixel). Depending on the format, the pixels can be represented in either binary and ascii. For our exercise, we prefer binary. However, we first have to take out the header to prevent the encryption from encoding the header. To do so, use your text editor (eg. vi, notepad) to take out the first two lines.

    ```
    $ cp org.pbm org.x
    $ vi org.x
    ```

    ~~P4~~

    ~~2000 2000~~

    KR)B@HD@H

    ```
    V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `V `W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _W _X!_X!_X!_X!_X!_X!_X!_X!_X!_X!_X!_X!_X!
    $□
    $□
    $□
    $□□$□□$□□$□□$□□$□□$□□$□□$□□$□
    $□
    $□
    $□
    $□
    $□
    $□
    $□
    $□
    %□
    %□
    %□
    %□
    %□
    %□
    %□
    %□
    &□
    &□
    $□  $□  $□  $□  $□  $□  $□  $□  $□  $□
    $□
    $□
    $□
    $□
    $□
    $□
    $□
    $□
    %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  %□  $□
    $□
    $□
    $□
    ```

[1] See wikipedia for more details. https://en.wikipedia.org/wiki/Netpbm

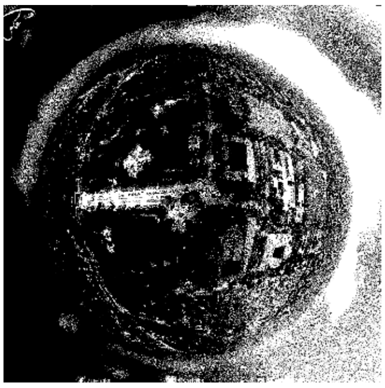
c. Encrypt the file with OpenSSL[1] with any block cipher algorithm in ECB mode (no padding and no salt).

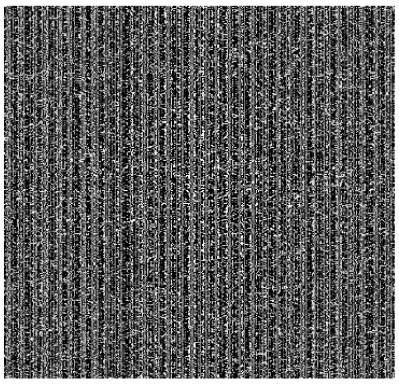
```
$ openssl enc -aes-256-ecb  -in org.x -nosalt \
      -out enc.x
```

d. Pad the header back and see the result.

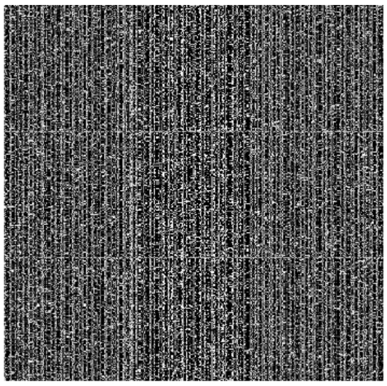
```
$ cp enc.x enc.pbm
$ vi enc.pbm
P4
2000 2000
KR)B@HD@
```

e. You may try it with other modes with IV, chaining, or feedback and compare the result.





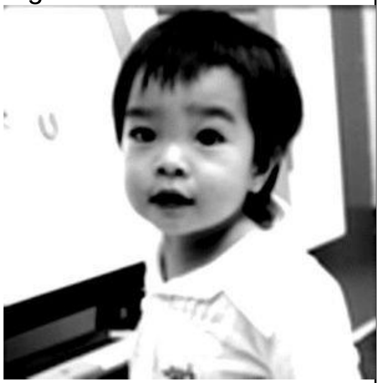
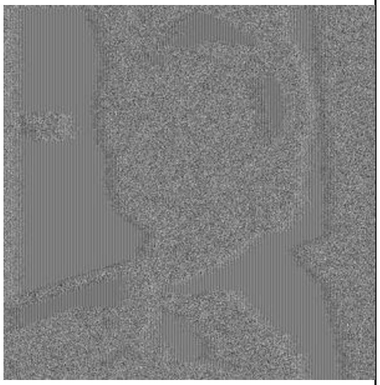
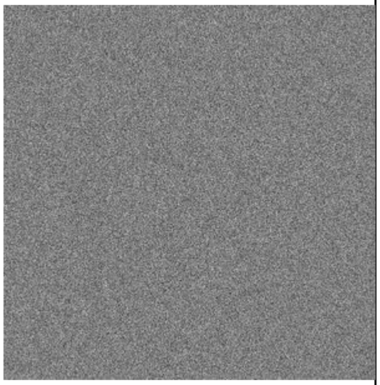
ecb



feedback

f. What does the result suggest about the mode of operation in block cipher? Please provide your analysis.
For my case the ecb and feedback is not much difference but in general you shouldn't use ecb because some pattern may still visible after encoding

If you got it all right, the result should be like this.

Original	AES-256-ECB	AES-256-CBC
		

[1] For details on command-line arguments, see https://wiki.openssl.org/index.php/Enc

18. (Encryption Protocol - Digital Signature)

a. Measure the performance of a hash function (sha1), RC4, Blowfish and DSA. Outline your experimental design.
(Please use OpenSSL for your measurement)
I write python to time the encryption of file with random string size 2kb
Time execution(ms)
Blow-fish(283) DSA(1023) RC4(315)

b. Comparing performance and security provided by each method.
perf: Blow-fish > RC4 > DSA
security blow-fish is block cipher
RC4 is steam cipher
the steam vs block is still a debating topic
DSA is slowest but strongest since it's asymmetric algo

c. Explain the mechanism underlying Digital Signature. How does it combine the strength and weakness of each encryption scheme?
digital signature let author sign fast with hash function and keep signature's integrity by using asymmetric algo

Hint: (OpenSSL command line)
# List algorithms

```
$ openssl list cipher-algorithms
# To encrypt
$ openssl enc -ciphername [options] -e -in filename -out filename \
     -K key -iv IV -nopad -nosalt
```