# COMP 524 - Cybersecurity: Assignment #1

Due on September 19th, 2018

*Dr. Michael Soltys*

Rihan Stephen Pereira
email - rihanstephen.pereira576@myci.csuci.edu, studentID - 002497665
David Bryne
email - david.byrne@myci.csuci.edu
Marco Cabrera
email - marco.cabrera561@myci.csuci.edu, studentID - 002456715
Kaveh Arashvand
email - kaveh.arashvand075@myci.csuci.edu, studentID - 002377350

September 20, 2018

# Problem 1

Decrypt the following four documents. To submit the assignment, write a two page PDF report, which explains, with code snippets, how you managed to decrypt the documents, and what the documents were.

- assignment-1-a.txt: Caesar cipher Base64 encoded text
- assignment-1-b.txt: MAC cipher Base64 encoded text
- assignment-1-c.txt: Caesar cipher Base64 encoded jpeg
- assignment-1-d.txt: MAC cipher Base64 encoded jpeg

**Solution:**
**Caesar cipher Base64 encoded text**
File - assignment-1-a.txt

A brute-force approach was used to solve this problem. From one of the three basic classical attacks, we applied *Ciphertext only attack*, also called as *recognizable plaintext attack*. Considering we are dealing with Caesar Cipher(CC) base64 encrypted text, we were sure that its using one of the 63 possible keys from 64 character alphabet list to encrypt the plaintext. From Dr. soltys notes, we used decryption function

$$D_n(x) = (x + n)(mod 64)$$

Also, we ignored keys $n = 0$ and $n = 64$, since the obtained ciphertext is same as plaintext. Each key was used to break the cipher followed by base64 decoding on the decrypted text to get the plaintext. On offset 19 i.e key $n = 19$, we got the plaintext. It is a poem from *William Blake*.

Python code snippet we used to obtain plaintext:

```
import base64

base64alphabet = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N',
                  'O','P','Q','R','S','T','U','V','W','X','Y','Z','a','b',
                  'c','d','e','f','g','h','i','j','k','l','m','n','o','p',
                  'q','r','s','t','u','v','w','x','y','z','0','1','2','3',
                  '4','5','6','7','8','9','+','/']

f = open("assignment-1-a-enc.txt", "r")
f.readline()
encrypted_base64_string = f.read()

for guess_offset in range(1, 64):
    base64_string = ""
    for character in encrypted_base64_string:
        if ((character != '=') and (character != '\n')):
            alphabet_index = base64alphabet.index(character)
            base64_string += base64alphabet[(alphabet_index -
                                             guess_offset) % 64]
        elif (character == '\n'):
            base64_string += '\n'
        else:
```

Problem 1 continued on next page...

```
        base64_string += '='
output = base64.b64decode(base64_string)
print(output)
```

The below images shows the plaintext that we obtained after running the above code fragment.

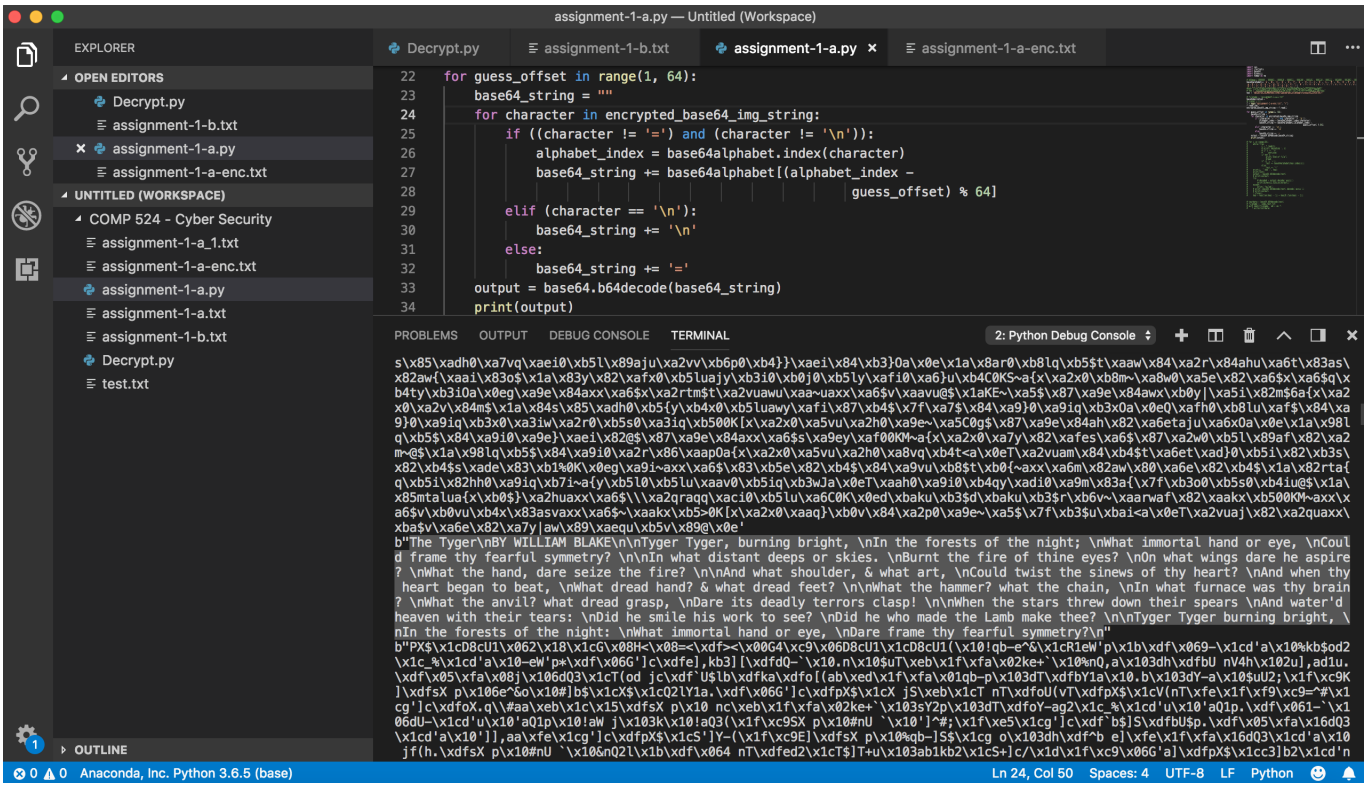


Figure 1: assignment-1-a.txt deciphered and decoded output

# Problem 2

**MAC cipher Base64 encoded text**
File - assignment-1-b.txt

For this problem, we tried to divide the base 64 encrypted text into 4 character chunks. Because, we know that each 3 character plaintext transfer to 4 letter cipher text. The biggest issue was that because of this transformation the location of each letter in base64 was important. We had to find a frequency function from the base64 to the plaintext.

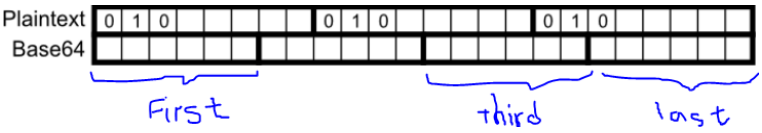


Figure 2: base 64 representation of 8 bit character

From our understanding of above figure, we created the below table which has the frequency of letters according to their locations in the ciphertext.

Now please consider table below: (TABLE OF FREQUENCY)

| All Alphabets in cipher | First Location in cipher | Third location in cipher | Last location in cipher |
|---|---|---|---|
| M: 197 **(important)** | Y:141 | V: 140 | l: 115 (lower el) |
| Q: 152 | U:128 | 2:79 | Q: 92 |
| U: 149 | I: 126 (capital i) | W:56 | e:65 |
| V: 144 | S: 107 | X:51 | F: 58 |
| Y:141 | /: 104 | L:51 | O: 57 (capital oh) |
| I: 132 (capital i) | Z:88 | Q:50 | z:47 |
| l:131 (lower el) | b:80 | F:47 | R:45 |
| S:129 | i: 36 | d:41 | x: 44 |
| /:110 | 5: 14 | T:38 | 6:43 |
| p: 108 | 2:13 | 0:37 (ZERO) | A:40 |
| F:105 | | s: 37 | 8:37 |
| m:102 | | r: 33 | P:34 |
| 2: 93 | | | D:30 |

Figure 3: Cipher frequency analysis

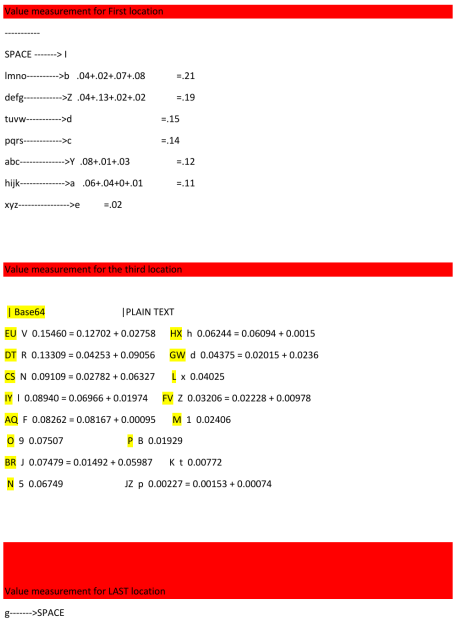


Figure 4: Frequency measurement value - Diagram 1

# Problem 3

**Caesar cipher Base64 encoded image**

File - assignment-1-c.txt

To crack this secret, we used *recognizable plaintext attack*, also called *Ciphertext only* attack. We performed brute force on the ciphertext - scanning all the keys and for each key we used the Decryption function

$$D_n(x) = (x + n)(mod64)$$

where, $x$ is the character from ciphertext and $n$ is the key for that iteration. Also, note that the key $n = 0$ or $n = 64$ were not considered because it doesn't not change the original plaintext. For each key, after running the decryption function, base64 decoding was applied to the text, and that base64 data was saved to disk as an image. Doing this, we saved altogether 63 images to the disk. Finally, we found key $n = 9$ was used to encrypt the plaintext which gave us the resulting image during decryption.

Python code snippet for decrypting and decoding Caesar Cipher base64 jpeg:

```
#! /usr/bin/python3

import mac
import os
import base64


def decrypt_cc_images():
```

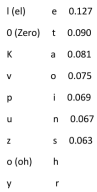


Figure 5: Frequency measurement value - Diagram 2

```
f = open("assignment-1-c-enc.txt", "r")
f.readline()
encrypted_base64_img_string = f.read()

for guess_offset in range(1, 64):
    base64_string = ""
    for character in encrypted_base64_img_string:
        if ((character != '=') and (character != '\n')):
            alphabet_index = mac.base64alphabet.index(character)
            base64_string += mac.base64alphabet[(alphabet_index -
                                                 guess_offset) % 64]
        elif (character == '\n'):
            base64_string += '\n'
        else:
            base64_string += '='

    temp_filename = "img_offset_" + str(guess_offset) + ".png"

    print('file %s written to disk'.format(temp_filename))
    img2disk = open(os.path.join("cc_base64_images", temp_filename),
                    "wb")
    img2disk.write(base64.b64decode(base64_string))
    img2disk.close()
```

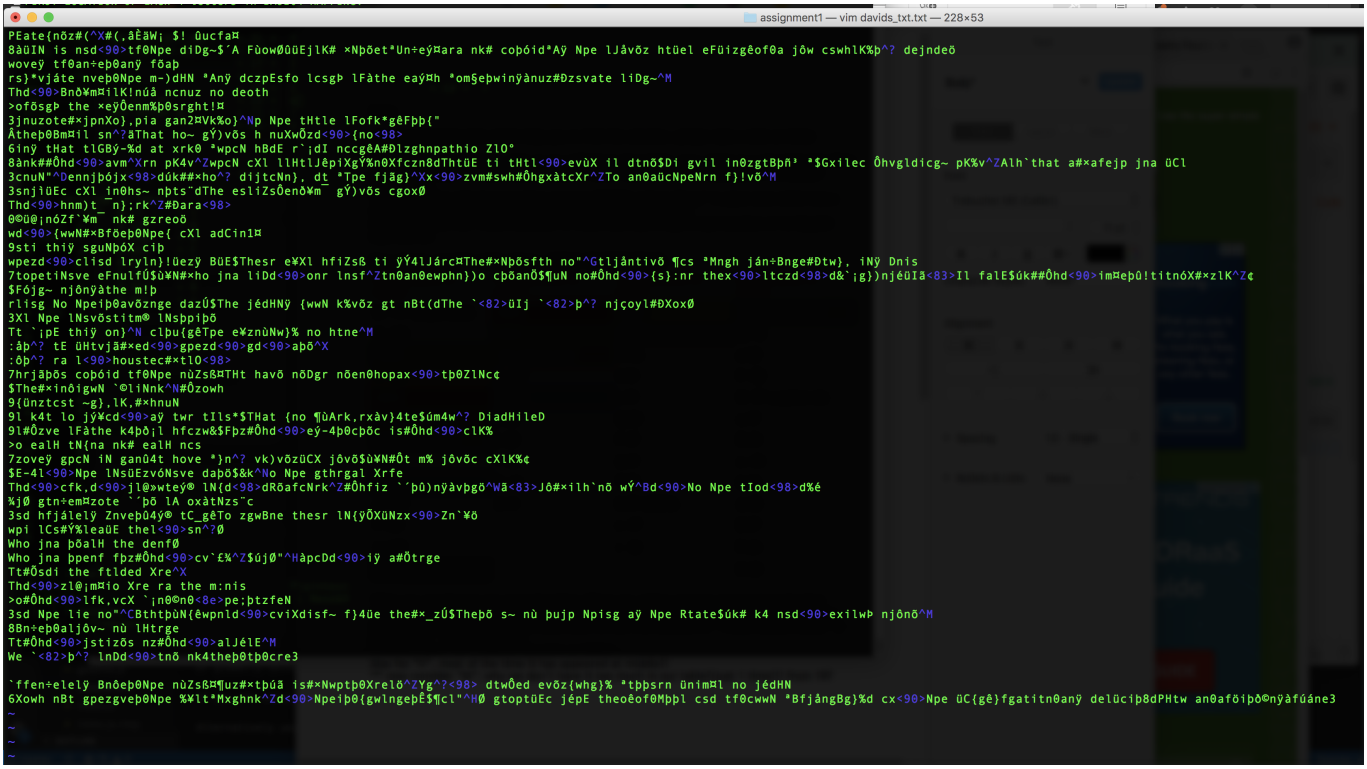


Figure 6: MAC Cipher partially decrypted base64 data

## `decrypt_cc_images()`

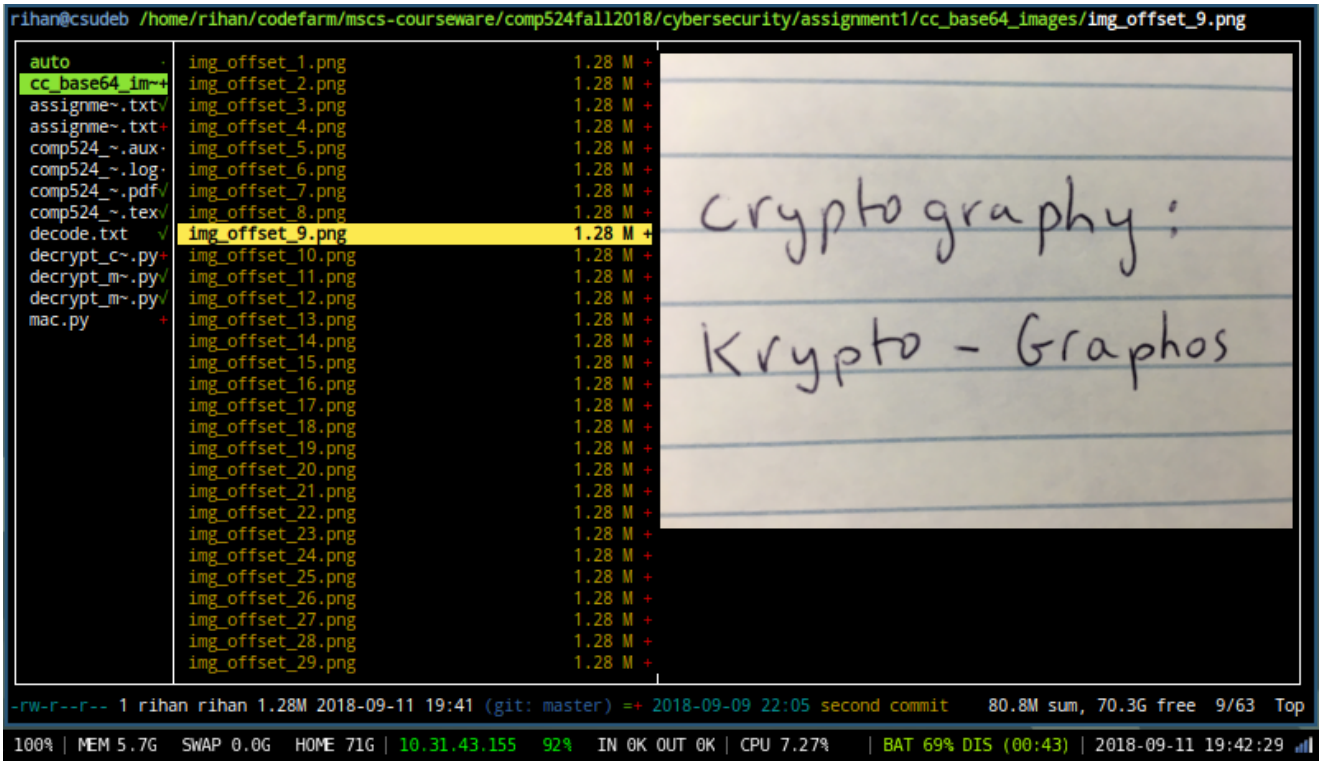


Figure 7: assignment-1-c.txt Caesar cipher decrypted and decoded image

# Problem 4

**MAC cipher Base64 encoded jpeg**
File - assignment-1-d.txt

The approach I used to solve this problem was a known text attack. By inspecting the other image that was decoded in problem 3 I was able to determine the header and byte offset that was included when the image was taken from an iphone. Then after we base64 encoded that known image I was able to count the exact offset of the header in base64 characters. Once I had this offset decrypting the key was just a matter of comparing the character at the same offset in both the encrypted file and the know file. Once we had a mapping for every character we were able to write a python script that decrypted and decoded the file.
The tool I used to calculate the byte offset of the image header was 0xd

The key is: 2JzvLTdIARk3nyDg9s0NhxjGf18YQrq/ei6ZM+ObWCKotaclw7FE5BXHV4puUmPS

The python program solution_1d.py will output the decrypted image to a file. You can run this code: python solution_1d.py

Here is the Python code snippet which uses above key to reveal the secret jpeg image.

```
import base64
base64alphabet = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N',
                  'O','P','Q','R','S','T','U','V','W','X','Y','Z','a','b',
                  'c','d','e','f','g','h','i','j','k','l','m','n','o','p',
                  'q','r','s','t','u','v','w','x','y','z','0','1','2','3',
```

Problem 4 continued on next page...

```
                    '4','5','6','7','8','9','+','/']

key="2JzvLTdIARK3nyDg9s0NhxjGf18YQrk/ei6ZM+ObWCqotaclw7FE5BXHV4puUmPS"
filename="assignment-1-d-enc-clean.txt"
image=""

with open(filename) as f:
  while True:
    c = f.read(1)
    print('decoding ',c)
    if not c:
      print("End of file")
      break
    if (c != '='):
      print(c,' ',key.index(c))
      image=image+base64alphabet[key.index(c)]
    else:
      image=image+"="

imgdata = base64.b64decode(image)
filename = 'solution_1d.jpg'  # I assume you have a way of picking
# unique filenames


with open(filename, 'wb') as f:
    f.write(imgdata)
```

The jpeg image secret that we found after breaking the ciphertext:

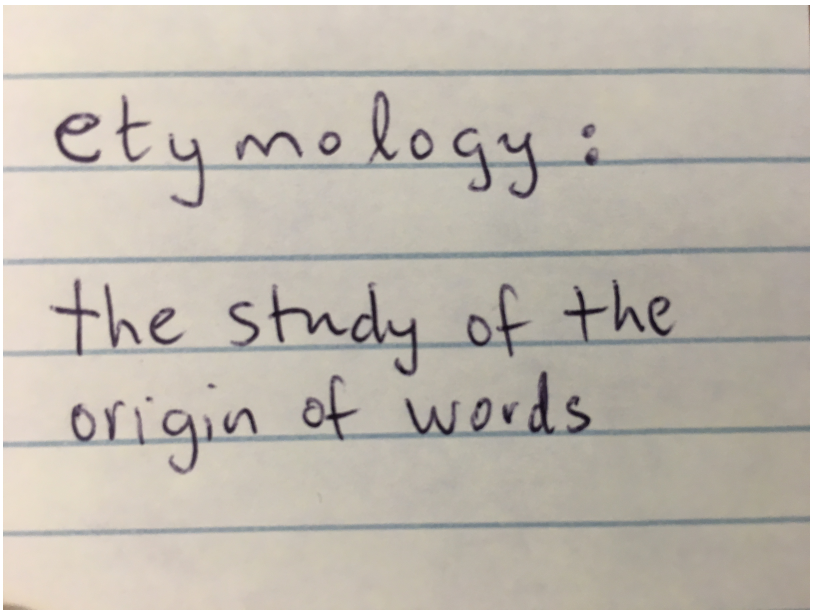


Figure 8: assignment-1-d.txt MAC cipher decrypted and decoded jpeg