Name: Shanqing Gu

Due 06/17/2018

MSDS 7349 Homework 2
Data and Network Security Homework Introduction to Ciphers

Exercise 1: Simple Substitution Cipher (30)

1. **Include the cipher.cy code (10 points)**

```
## codes for Q1-2

def encrypt(test, s):
    encoded_phrase = ""

    # traverse text
    for i in range(len(phrase)):
        char = phrase[i]

        # Encrypt uppercase characters
        if (char.isupper()):
            encoded_phrase += "X"

        # Encrypt lowercase characters
        elif (char.islower()):
            encoded_phrase += "x"
        else:
            encoded_phrase += chr((ord(char)))

    return encoded_phrase

# check the above function
phrase = "Mayday! Mayday!"
s = 4
print("Enter sentence to encrypt: " + phrase)
print("Enter shift value: " + str(s))
print("The encoded phrase is: " + encrypt(phrase,s))


## codes for Q1-3

def encrypt(test, s):
    encoded_phrase = ""

    for i in range(len(phrase)):
        char = phrase[i]

        if (char.isupper()):
            encoded_phrase += chr((ord(char) + s-65) % 26 + 65)

        elif (char.islower()):
            encoded_phrase += chr((ord(char) + s - 97) % 26 + 97)
        else:
            encoded_phrase += chr((ord(char)))

    return encoded_phrase
```

```
phrase = "Mayday! Mayday!"
s = 4

print("Enter sentence to encrypt: " + phrase)
print("Enter shift value: " + str(s))
print("The encoded phrase is: " + encrypt(phrase,s))
```

2. Include screen shoot for your answer for part 1-2 (10 points)

```
## 7349_hw2: Simple Substitution Cipher

def encrypt(test, s):
    encoded_phrase = ""

    # traverse text
    for i in range(len(phrase)):
        char = phrase[i]

        # Encrypt uppercase characters
        if (char.isupper()):
            encoded_phrase += "X"

        # Encrypt lowercase characters
        elif (char.islower()):
            encoded_phrase += "x"
        else:
            encoded_phrase += chr((ord(char)))

    return encoded_phrase

# check the above function
phrase = "Mayday! Mayday!"
s = 4
print("Enter sentence to encrypt: " + phrase)
print("Enter shift value: " + str(s))
print("The encoded phrase is: " + encrypt(phrase,s))
```

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is: Xxxxxx! Xxxxxx!
```

3. Include screen shoot for your answer for part 1-3 (10 Points)

```
## 7349_hw2: Simple Substitution Cipher

def encrypt(test, s):
    encoded_phrase = ""

    # traverse text
    for i in range(len(phrase)):
        char = phrase[i]

        # Encrypt uppercase characters
        if (char.isupper()):
            encoded_phrase += chr((ord(char) + s-65) % 26 + 65)

        # Encrypt lowercase characters
        elif (char.islower()):
            encoded_phrase += chr((ord(char) + s - 97) % 26 + 97)
        else:
            encoded_phrase += chr((ord(char)))

    return encoded_phrase

# check the above function
phrase = "Mayday! Mayday!"
s = 4
print("Enter sentence to encrypt: " + phrase)
print("Enter shift value: " + str(s))
print("The encoded phrase is: " + encrypt(phrase,s))
```

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is: Qechec! Qechec!
```

Exercise 2: Breaking a Simple Substitution Cipher (30)

1. Include the program code (20 points)

```
## codes for Q2

def decrypt(phrase):
    for s in range (1,27):
        deciphered_phrase = ""

    # traverse text
        for i in range(len(phrase)):
            char = phrase[i]

        # decrypt uppercase characters
            if (char.isupper()):
               deciphered_phrase += chr((ord(char) + s-65) % 26 + 65)

        # decrypt lowercase characters
            elif (char.islower()):
                deciphered_phrase += chr((ord(char) + s - 97) % 26 + 97)
            else:
                deciphered_phrase += chr((ord(char)))
        print(s,deciphered_phrase)

    return deciphered_phrase

decrypt(phrase="Qtm, Enqqdrs, Qtm! Mnv xnt vntkcm's adkhdud ld he H snkc xnt ats H bzm
qtm khjd sgd vhmc aknvr.")
```

2. Include a screen capture of the program in action and the posted cipher text and its corresponding plaintext and key as retrieved by your program for each of the posted cipher texts (10 points)

```python
## 7349_hw2_exercise 2: Breaking a simple substituion cipher

def decrypt(phrase):
    for s in range (1,27):
        deciphered_phrase = ""

    # traverse text
        for i in range(len(phrase)):
            char = phrase[i]

        # decrypt uppercase characters
            if (char.isupper()):
               deciphered_phrase += chr((ord(char) + s-65) % 26 + 65)

        # decrypt lowercase characters
            elif (char.islower()):
                deciphered_phrase += chr((ord(char) + s - 97) % 26 + 97)
            else:
                deciphered_phrase += chr((ord(char)))
        print(s,deciphered_phrase)

    return deciphered_phrase

decrypt(phrase="Qtm, Enqqdrs, Qtm! Mnv xnt vntkcm's adkhdud ld he H snkc xnt ats H bzm qtm khjd sgd vhmc aknvr.")
```

```
1 Run, Forrest, Run! Now you wouldn't believe me if I told you but I can run like the wind blows.
2 Svo, Gpssftu, Svo! Opx zpv xpvmeo'u cfmjfwf nf jg J upme zpv cvu J dbo svo mjlf uif xjoe cmpxt.
3 Twp, Hqttguv, Twp! Pqy aqw yqwnfp'v dgnkgxg og kh K vqnf aqw dwv K ecp twp nkmg vjg ykpf dnqyu.
4 Uxq, Iruuhvw, Uxq! Qrz brx zrxogq'w eholhyh ph li L wrog brx exw L fdq uxq olnh wkh zlqg eorzv.
5 Vyr, Jsvviwx, Vyr! Rsa csy asyphr'x fipmizi qi mj M xsph csy fyx M ger vyr pmoi xli amrh fpsaw.
6 Wzs, Ktwwjxy, Wzs! Stb dtz btzqis'y gjqnjaj rj nk N ytqi dtz gzy N hfs wzs qnpj ymj bnsi gqtbx.
7 Xat, Luxxkyz, Xat! Tuc eua cuarjt'z hkrokbk sk ol O zurj eua haz O igt xat roqk znk cotj hrucy.
8 Ybu, Mvyylza, Ybu! Uvd fvb dvbsku'a ilsplcl tl pm P avsk fvb iba P jhu ybu sprl aol dpuk isvdz.
9 Zcv, Nwzzmab, Zcv! Vwe gwc ewctlv'b jmtqmdm um qn Q bwtl gwc jcb Q kiv zcv tqsm bpm eqvl jtwea.
10 Adw, Oxaanbc, Adw! Wxf hxd fxdumw'c knurnen vn ro R cxum hxd kdc R ljw adw urtn cqn frwm kuxfb.
11 Bex, Pybbocd, Bex! Xyg iye gyevnx'd lovsofo wo sp S dyvn iye led S mkx bex vsuo dro gsxn lvygc.
12 Cfy, Qzccpde, Cfy! Yzh jzf hzfwoy'e mpwtpgp xp tq T ezwo jzf mfe T nly cfy wtvp esp htyo mwzhd.
13 Dgz, Raddqef, Dgz! Zai kag iagxpz'f nqxuqhq yq ur U faxp kag ngf U omz dgz xuwq ftq iuzp nxaie.
14 Eha, Sbeerfg, Eha! Abj lbh jbhyqa'g oryvrir zr vs V gbyq lbh ohg V pna eha yvxr gur jvaq oybjf.
15 Fib, Tcffsgh, Fib! Bck mci kcizrb'h pszwsjs as wt W hczr mci pih W qob fib zwys hvs kwbr pzckg.
16 Gjc, Udggthi, Gjc! Cdl ndj ldjasc'i qtaxtkt bt xu X idas ndj qji X rpc gjc axzt iwt lxcs qadlh.
17 Hkd, Vehhuij, Hkd! Dem oek mekbtd'j rubyulu cu yv Y jebt oek rkj Y sqd hkd byau jxu mydt rbemi.
18 Ile, Wfiivjk, Ile! Efn pfl nflcue'k svczvmv dv zw Z kfcu pfl slk Z tre ile czbv kyv nzeu scfnj.
19 Jmf, Xgjjwkl, Jmf! Fgo qgm ogmdvf'l twdawnw ew ax A lgdv qgm tml A usf jmf dacw lzw oafv tdgok.
20 Kng, Yhkkxlm, Kng! Ghp rhn phnewg'm uxebxox fx by B mhew rhn unm B vtg kng ebdx max pbgw uehpl.
21 Loh, Zillymn, Loh! Hiq sio qiofxh'n vyfcypy gy cz C nifx sio von C wuh loh fcey nby qchx vfiqm.
22 Mpi, Ajmmzno, Mpi! Ijr tjp rjpgyi'o wzgdzqz hz da D ojgy tjp wpo D xvi mpi gdfz ocz rdiy wgjrn.
23 Nqj, Bknnaop, Nqj! Jks ukq skqhzj'p xaheara ia eb E pkhz ukq xqp E ywj nqj hega pda sejz xhkso.
24 Ork, Cloobpq, Ork! Klt vlr tlriak'q ybifbsb jb fc F qlia vlr yrq F zxk ork ifhb qeb tfka yiltp.
25 Psl, Dmppcqr, Psl! Lmu wms umsjbl'r zcjgctc kc gd G rmjb wms zsr G ayl psl jgic rfc uglb zjmuq.
26 Qtm, Enqqdrs, Qtm! Mnv xnt vntkcm's adkhdud ld he H snkc xnt ats H bzm qtm khjd sgd vhmc aknvr.

"Qtm, Enqqdrs, Qtm! Mnv xnt vntkcm's adkhdud ld he H snkc xnt ats H bzm qtm khjd sgd vhmc aknvr."
```

Brandon:
Encrypted: Nk dtz bfsy yt bfqp ymj mjfajsqd xywjjyx tk ltqi, dtz ltyyf pstb ymj ufxxbtwi, “Wtqq, Ynij, Wtqq!”
Decrypted: If you want to walk the heavenly streets of gold, you gotta know the password, 'Roll, Tide, Roll!'

Lizzy:
Encrypted: Hexe erh Rixasvo Wigyvmxc!
Decrypted: Data and Network Security!

Noelle:
Encrypted: Jgnnq. Oa pcog ku Kpkiq Oqpvqac. Aqw mkngf oa hcvjgt. Rtgrctg vq fkg.
Decrypted: Hello. My name is Inigo Montoya. You killed my father. Prepare to die.

Brett:
Encrypted: "Tmetgxtcrt xh iwt itprwtg du paa iwxcvh." - Yjaxjh Rpthpg
Decrypted: 'Experience is the teacher of all things.' - Julius Caesar

Ann:
Encrypted: Yzestyr mftwod dpwq-pdeppx lyo dpwq-nzyqtopynp wtvp lnnzxawtdsxpye.
Decrypted: Nothing builds self-esteem and self-confidence like accomplishment.

Shanqing:
Encrypted: Qtm, Enqqdrs, Qtm! Mnv xnt vntkcm's adkhdud ld he H snkc xnt ats H bzm qtm khjd sgd vhmc aknvr.
Decrypted: Run, Forrest, Run! Now you wouldn't believe me if I told you but I can run like the wind blows.

Kyle:
Encrypted: Bx cqn lxvkrwjcrxw rb... xwn, cfx, cqann, oxda, oren? Cqjc'b cqn bcdyrmnbc lxvkrwjcrxw R'en nena qnjam rw vh uron! Cqjc'b cqn trwm xo cqrwp jw rmrxc fxdum qjen xw qrb udppjpn!

Decrypted: So the combination is... one, two, three, four, five? That's the stupidest combination I've ever heard in my life! That's the kind of thing an idiot would have on his luggage!

Lu:
Encrypted: Qtm, Enqqdrs, Qtm! Mnv xnt vntkcm's adkhdud ld he H snkc xnt ats H bzm qtm khjd sgd vhmc aknvr.
Decrypted: Nature doesn't recognize good and evil. Nature only recognizes balance and imbalance.

Michael:
Encrypted: Hkh E gjks ukq qoaz xnqpa bknya, pda gau eo pkpwhhu psajpu psk.
Decrypted: Lol I know you used brute force, the key is totally twenty two.

Students have not posted until this submission: Vanessa, Ryan, Travis, Scott, and Alexandra

Exercise 3: AES (40 points)

1. Include the program code (10 points)

```
## codes for Q3 AES

from IPython.display import display
from PIL import Image
from Crypto.Cipher import AES

filename = "/Users/shanqinggu/Desktop/rose.jpeg"
filename_out = "/Users/shanqinggu/Desktop/rose_encrypted"

format = "jpeg"
key = "abcdabcdabcdabcd"

# AES requires that plaintexts be a multiple of 16, so we have to pad the data
def pad(data):
    return data + b"\x00"*(16-len(data)%16)

# Maps the RGB
def convert_to_RGB(data):
    r, g, b = tuple(map(lambda d: [data[i] for i in range(0,len(data)) if i % 3 == d], [0, 1, 2]))
    pixels = tuple(zip(r,g,b))
    return pixels

def process_image(filename):
    # Opens image and converts it to RGB format for PIL
    im = Image.open(filename)
    data = im.convert("RGB").tobytes()

    # Pad the data to satisfy AES's multiple-of-16 requirement
    original = len(data)

    # Encrypts using desired AES mode (ECB by default, and change to CBC or GCM by selection)
    new = convert_to_RGB(aes_ecb_encrypt(key, pad(data))[:original])

    # Create a new PIL Image object and save the old image data into the new image.
    im2 = Image.new(im.mode, im.size)
    im2.putdata(new)

    # Save image
    im2.save(filename_out+"."+format, format)
    return
```

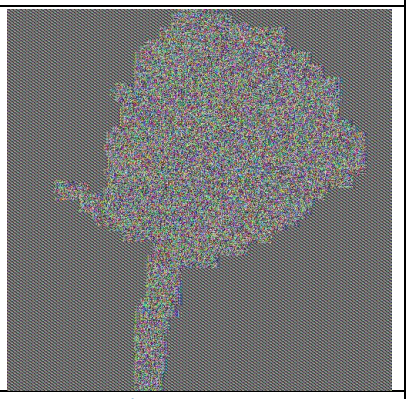
```
# ECB mode by default
def aes_ecb_encrypt(key, data, mode=AES.MODE_ECB):
    aes = AES.new(key.encode("utf8"), mode)
    new_data = aes.encrypt(data)
    return new_data

# CBC mode by selection
def aes_cbc_encrypt(key, data, mode=AES.MODE_CBC):
    IV = "B"*16  #We'll manually set the initialization vector to simplify things
    aes = AES.new(key.encode("utf8"), mode, IV.encode("utf8"))
    new_data = aes.encrypt(data)
    return new_data

# Counter Mode by selection
def aes_gcm_encrypt(key, data, mode=AES.MODE_GCM):
    aes = AES.new(key.encode("utf8"), mode)
    new_data = aes.encrypt(data)
    return new_data

process_image(filename)
```

2. Include a screen capture of the program in action, the original jpg and each of the encrypted versions of the jpg. (30 Points)

Original image	ECB mode
	
CBC mode	Counter Mode