Advanced Security Assignment 1

1. Some methods of securing your web browser include:
   - Ad blockers: Ad blockers can be used to block data and tracking tools.
   - VPNs: VPNs provide a layer of security to your browser by creating a connection between you and the internet which is secure. It does this by using an encrypted virtual tunnel to route your data through, encrypting your data and hiding your IP address.
   - Antivirus: This protects your system from being infected with malware and trojans, by scanning websites for malicious files.
   - HTTPS: This is a protocol which encrypts communication between your browser and the website you are connected to, using an asymmetric encryption.

   There are various reasons as to why we are being tracked. For video streaming sites such as YouTube and Netflix, it is for their algorithm to work to recommend movies you would like based on your watch history. Similarly, shopping websites would do it to recommend products you may like to purchase. Search engines might use your data to recommend ads to you of items you previously searched.

   Without being 100% offline, it is hard to know whether it is possible to fully protect yourself from being tracked, however some of the methods I mentioned above are great ways of minimizing it and ensuring you are protected online. As well as these methods, one must be sure to do things such as disable social media tracking, to not allow cookies, alter their privacy and permissions settings accordingly and using private browsing tabs.

2. **Caesar cipher** encryption and decryption program:

```
#program to encrypt message
def encryptText(text,key):
    text = text.upper()

    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

    result = ''

    for char in text:
        if char in alphabet:
            index = (alphabet.find(char) + key) % len(alphabet)
            result = result + alphabet[index]

        else:
            result = result + char

    return result

#program to decrypt message
def decryptText(text,key):
    text = text.upper()

    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

    result = ''

    for char in text:
        if char in alphabet:
            index = (alphabet.find(char) - key) % len(alphabet)
            result = result + alphabet[index]

        else:
            result = result + char

    return result

#check the above function
```

```
key = 4
text = "texttoencrypt"
print (encryptText(text,key))
dtext = encryptText(text,key)
print(decryptText(dtext,key))
```

Vigenère Cipher

```python
#generates key
def createKey(text, key):
    newkey = ''
    i = 0
    for char in text:
        if char.isalpha():
            newkey += key[i % len(key)]
            i += 1
        else:
            newkey += ' '
    return newkey

# get the position of each character
def getCharPosition(textChar, keyChar, action):
    if textChar.isalpha():
        textChar = textChar.upper()
        firstLetter = 'A'

        charPosition = ord(textChar) - ord(firstLetter)
        keyCharPosition = ord(keyChar) - ord('a')

        if action == 'encrypt':
            cipherChar = (charPosition + keyCharPosition) % 26

        else:
            cipherChar = (charPosition - keyCharPosition) % 26

        return chr(cipherChar + ord(firstLetter))
    return textChar

# encrypt text
def encrypt(text, key):
    encryptedText = ''
    encryptKey = createKey(text, key)
    for textc, keyc in zip(text, encryptKey):
        encryptedText += getCharPosition(textc, keyc, 'encrypt')
    return encryptedText
```

```
# decrypt text
def decrypt(encryptedText, key):
    decryptedText = ''
    decryptKey = createKey(encryptedText, key)
    for encryptedTextc, keyc in zip(encryptedText, decryptKey):
        decryptedText += getCharPosition(encryptedTextc, keyc, 'decrypt')
    return decryptedText

text = 'TEXTTOENCRYPT'
key = 'LEMON'

encrypted = encrypt(text, key)
print(encrypted)
print(decrypt(encrypted, key))
```

3. ONE VARIATION TO THE STANDARD CAESAR CIPHER IS WHEN THE ALPHABET IS "KEYED" BY USING A WORD. IN THE TRADITIONAL VARIETY, ONE COULD WRITE THE ALPHABET ON TWO STRIPS AND JUST MATCH UP THE STRIPS AFTER SLIDING THE BOTTOM STRIP TO THE LEFT OR RIGHT. TO ENCODE, YOU WOULD FIND A LETTER IN THE TOP ROW AND SUBSTITUTE IT FOR THE LETTER IN THE BOTTOM ROW. FOR A KEYED VERSION, ONE WOULD NOT USE A STANDARD ALPHABET, BUT WOULD FIRST WRITE A WORD (OMITTING DUPLICATED LETTERS) AND THEN WRITE THE REMAINING LETTERS OF THE ALPHABET. FOR THE EXAMPLE BELOW, I USED A KEY OF "RUMKIN.COM" AND YOU WILL SEE THAT THE PERIOD IS REMOVED BECAUSE IT IS NOT A LETTER. YOU WILL ALSO NOTICE THE SECOND "M" IS NOT INCLUDED BECAUSE THERE WAS AN M ALREADY AND YOU CAN'T HAVE DUPLICATES

4. Key = 17 shifts to the right OR 9 shifts to the left

5. NIST IS ABOUT TO ANNOUNCE THE NEW HASH ALGORITHM THAT WILL BECOME SHA-3. THIS IS THE RESULT OF A SIX-YEAR COMPETITION, AND MY OWN SKEIN IS ONE OF THE FIVE REMAINING FINALISTS (OUT OF AN INITIAL 64). IT'S PROBABLY TOO LATE FOR ME TO AFFECT THE FINAL DECISION, BUT I AM HOPING FOR "NO AWARD." IT'S NOT THAT THE NEW HASH FUNCTIONS AREN'T ANY GOOD, IT'S THAT WE DON'T REALLY NEED ONE. WHEN WE STARTED THIS PROCESS BACK IN 2006, IT LOOKED AS IF WE WOULD BE NEEDING A NEW HASH FUNCTION SOON. THE SHA FAMILY (WHICH IS REALLY PART OF THE MD4 AND MD5 FAMILY), WAS UNDER INCREASING PRESSURE FROM NEW TYPES OF CRYPTANALYSIS. WE DIDN'T KNOW HOW LONG THE VARIOUS SHA-2 VARIANTS WOULD REMAIN SECURE. BUT IT'S 2012, AND SHA-512 IS STILL LOOKING GOOD

6.

2*2 Hill Cipher

```python
import numpy as np


def encrypt(text, key):
    text = text.replace(" ", "")
    # if message length is an odd number add a zero at the end
    length_count = 0
    if len(text) % 2 != 0:
        text += "0"
        length_count = 1

    # convert text to matrix
    row = 2
    column = int(len(text)/2)
    textToD = np.zeros((row, column), dtype=int)
    it1 = 0
    it2 = 0
    for i in range(len(text)):
        if i % 2 == 0:
            textToD[0][it1] = int(ord(text[i])-65)
            it1 += 1
        else:
            textToD[1][it2] = int(ord(text[i])-65)
            it2 += 1
    key = key.replace(" ", "")

    # convert key to matrix
    keyToD = np.zeros((2, 2), dtype=int)
    it3 = 0
    for i in range(2):
        for j in range(2):
            keyToD[i][j] = ord(key[it3])-65
            it3 += 1

    # check key is valid and find determinant
    determinant = keyToD[0][0] * keyToD[1][1] - keyToD[0][1] * keyToD[1][0]
    determinant = determinant % 26
```

```
    # find multiplicative inverse
    m_inverse = -1
    for i in range(26):
        tmpInv = determinant * i
        if tmpInv % 26 == 1:
            m_inverse = i
            break
        else:
            continue

    if m_inverse == -1:
        print("Invalid key")
        sys.exit()



    encryptedTxt = ""
    count = int(len(text)/2)
    if length_count == 0:
        for i in range(count):
            temp1 = textToD[0][i] * keyToD[0][0] + textToD[1][i] * keyToD[0][1]
            encryptedTxt += chr((temp1 % 26) + 65)
            temp2 = textToD[0][i] * keyToD[1][0] + textToD[1][i] * keyToD[1][1]
            encryptedTxt += chr((temp2 % 26) + 65)
    else:
        for i in range(count-1):
            temp1 = textToD[0][i] * keyToD[0][0] + textToD[1][i] * keyToD[0][1]
            encryptedTxt += chr((temp1 % 26) + 65)
            temp2 = textToD[0][i] * keyToD[1][0] + textToD[1][i] * keyToD[1][1]
            encryptedTxt += chr((temp2 % 26) + 65)

    print("Encrypted Text: {}".format(encryptedTxt))


text = input("Enter plaintext to encrypt: ").upper()
key = input("Enter a four letter Key: ").upper()

encrypt(text, key)
```