MSDS 7349
Data and Network Security
Homework
Introduction to Ciphers

Due Week 12

Name:

What to Submit

Your final submission shall be a single pdf document that includes this document, screen captures of your exercises plus your answers to each of the written questions (if any). Note that you are expected to clearly label each section so as to make it clear to the instructor what files and data belong to which exercise/question. In your submitted pdf document, place your last name and the homework number at the beginning of the file name. For example, the filename for Homework 2 for Daniel Engels should be *EngelsHW2MSDS7349.pdf*.

Collaboration is expected and encouraged; however, each student must hand in their own homework assignment. To the greatest extent possible, answers should not be copied but, instead, should be written in your own words. Copying answers from anywhere is plagiarism, this includes copying text directly from the textbook. Do not copy answers. Always use your own words. For each question list all persons with whom you collaborated and list all resources used in arriving at your answer. Resources include but are not limited to the textbook used for this course, papers read on the topic, and Google search results. Note that 'Google' is not a resource. Don't forget to place your name on the document.

Exercise 1 : Simple Substitution Cipher

The goal of this exercise is to write a cyclic cipher to encrypt messages. This type of cipher was used by Julius Caesar to communicate with his generals. It is very simple to generate but it can actually be easily broken and does not provide the security one would hope for.

The key idea behind the Caesar cipher is to replace each letter by a letter some fixed number of positions down the alphabet. For example, if we want to create a cipher shifting by 3, you will get the following mapping:

```
Plain text: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext: DEFGHIJKLMNOPQRSTUVWXYZABC
```

To be able to generate the cipher above, we need to understand a little bit about how text is represented inside the computer. Each character has a numerical value and one of the standard encodings is ASCII (American Standard Code for Information Interchange). It is a mapping between the numerical value and the character graphic. For example, the ASCII value of 'A' is 65 and the ASCII value of 'a' is 97. To convert between the ASCII code and the character value in Python, you can use the following code:

```
letter = a
# converts a letter to ascii code asciiCode = ord(letter)
# converts ascii code to a letter letterRes = chr(asciiCode)
print ascii_code, letter_res
```

Start small. Do not try to implement the entire program at once. Break the program into parts as follows:

1) Create a file called cipher.py. Start your program by asking the user for a phrase to encode and the shift value. Then begin the structure of your program by entering in this loop (we'll build on it more in a bit):

   ```
   encodedPhrase = ""
   for c in phrase:
   encodedPhrase = encodedPhrase + c
   ```

   What does this loop do? Make sure you understand what the code does before moving on!
2) Now modify the program above to replace all the alphabetic characters with 'x'. For example:

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

We are going to apply the cipher only to the alphabetic characters and we will ignore the others.

3) Now modify your code, so that it produces the encoded string using the cyclic cipher with the shift value entered by the user. Let's see how one might do a cyclic shift. Let's say we have the sequence:

   ```
   012345
   ```

   If we use a shift value of 4 and just shift all the numbers, the result will be:

   ```
   456789
   ```

   We want the values of the numbers to remain between 0 and 5. To do this we will use the modulus operator. The expression x%y will return a number in the range 0 to y-1 inclusive, e.g. 4%6 = 4, 6%6 = 0, 7%6 =1. Thus the result of the operation will be:

   ```
   450123
   ```

   Hint: Note that the ASCII value of 'A' is 65 and 'a' is 97, not 0. So you will have to think how to use the modulus operator to achieve the desired result.
   Apply the cipher separately to the upper and lower case letters.
   Here is what you program should output:

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

Turn in a copy of the program and a screen capture of the program in action.

Exercise 2 : Breaking a Simple Substitution Cipher

The goal of this exercise is to retrieve the key and plaintext used to create a given cipher text string. For this exercise, write an algorithm that works in better than brute force time to retrieve the key and plaintext where the input is a cipher text string as generated by your cipher from the previous exercise.

Each student must post an encrypted message to the course Wall, and each student must retrieve the message encrypted by each of the posts on the Wall.

Turn in a copy of the program, 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.

Exercise 3 : AES

Now that we have some experience with simple symmetric key ciphers, let's get some basic experience with the most widely used standardized cipher in the world today: AES (Advanced Encryption Standard).

AES was standardized by NIST in U.S. FIPS PUB 197 (FIPS 197) in 2001 after a lengthy competition. AES has stood the test of time and cryptographic analysis to provide cryptographically strong functionality.

The goal of this exercise is to write (or find and heavily document) an AES 128 bit Python implementation and then to compare the performance of three different modes of operation using AES to encrypt and render on the display a large jpg file.

First, either write or obtain a Python implementation of AES-128. Then implement the following three modes of operation for AES: ECB, CBC and Counter Mode. Perform an encryption using each of these three modes on a jpg file of your choosing.

After encryption, render on your display the results of the encryption.

Turn in a copy of the program, a screen capture of the program in action, the original jpg and each of the encrypted versions of the jpg.