Data Privacy & Security Through Cryptography

Due: Tuesday, July 20th by 10:30am (before class)

Description:

During Dr. Daniel Votipka's presentation last week you were introduced to the importance of data privacy and security, as well as some associated technologies that we use every day like HTTPS and TLS. In this lab, you will learn more about the fields of *cryptography* and *cryptanalysis*.

Part 1, Cryptography:

In simplest terms, cryptography is the study and practice of techniques for secure communication and data storage in the presence of malicious third parties, often referred to as "adversaries". Until recently, cryptography almost solely referred to the processes of *encryption* and (the reverse) *decryption*, which involves converting data (plaintext) into an unintelligible format (ciphertext). This process is performed by a pair of algorithms which, together, are referred to as a *cipher*. A cipher uses one or more "secret keys" to perform the encryption and decryption processes – hence the keys are known only to trusted parties. Some examples of widely used ciphers that you may recognize include RSA, AES, and DES.

In this part of the lab, you will implement one of the most basic and earliest known ciphers, the *Caesar cipher*. The Caesar cipher works by simply replacing each letter in the message by a letter located a fixed number of positions down the alphabet – AKA 'shifting' the letter by the key value. For example, the following plaintext/ciphertext pairs demonstrate a Caesar cipher with a key value of 1:

Plaintext: PYTHON IS PRETTY COOL HUH?
Ciphertext: QZUIPO JT QSFUUZ DPPM IVI?

Your first tasks are to fill in the starter code for the encryption and decryption functions. Some things to keep in mind are: 1) make sure that applying the alphabetic shift to a letter with the key rotates back to the beginning of the alphabet, if necessary (i.e., Z + 1 = A), 2) Python has functions to convert to lower/uppercase and between numeric and character ascii values, and 3) do not apply the alphabetic shifting to non-alphabetic characters like (?,!,",1235, etc.).

To start off, you may wish to implement the functions while converting the plaintext to all uppercase or lowercase for simplicity. **Consider maintaining the case of every letter to be extra credit**.

(Extra Credit!!) Part 2, Cryptanalysis:

Cryptanalysis is the practice of analyzing cryptographic algorithms and implementations for weaknesses. It is used to breach security systems to gain access to encrypted messages. So, you could compare cryptanalysis with exciting

Hollywood terms like "**hacking**" and "**codebreaking**" 😊. In general, cryptanalysis is very important to validate the strength of the methods we use to keep ourselves safe on the internet.

In this part of the lab, you will mount a statistical attack on the Caesar cipher you implemented earlier. While a brute-force search would be simple because there are only 26 possible key values, the statistical attack will demonstrate the inherent weakness of *all* alphabetic ciphers.

In the starter code function statistical_attack() you are provided a list of decimals called unigramFreqs. These frequencies represent the commonality of that letter in words in the English language. For instance, the letter 't' has a frequency of 0.09, and 'z' of 0.002. You will use these to compute a correlation value for each possible key that can be used for decryption. The process is outlined as follows:

1. Compute frequency *f(c)* of each letter *c* in ciphertext,
   where *f(c)* = $\frac{\#\ of\ occurrences\ of\ each\ letter}{total\ \#\ of\ letters}$
2. Apply 1-gram model of English – compute the correlation (φ) of frequency of letters in the *ciphertext* with frequency of corresponding letters in *English* for key *i,* formalized as:
   a. For key i:
   $$\varphi(\mathrm{i}) = \sum_{c=0}^{25} f(c) * unigramFreqs(c-i)$$
   with the intuition being that φ(i) represents the sum of probabilities for words in the plaintext if *i* were the key.

Using this approach, you will implement the statistical attack and derive the value for the key. The index (i) for the largest correlation value you compute should correspond to the correct key value.

Files Given:

Lab-06 in Replit:
main.py – Starter code to help you begin building your solution to the exercise.
tufts_fight_song.txt – Tufts' fight song text file which you will encrypt and decrypt.

Helpful Resources:

- https://en.wikipedia.org/wiki/Caesar_cipher
- https://www.asciitable.com/
- https://www.askpython.com/python/built-in-methods/python-chr-and-ord-methods

How to submit your lab:

Submit your lab directly through Replit before the next day of class. (Morning session has an extra day to finish).