Introduction to Cryptography
Final Project

## 1. Classical Encryption

(1) Rail Fence Cipher: Write functions in Python to encrypt and decrypt text messages using the so-called "Rail Fence Cipher." First write functions in the case of 2, 3, and 4 lines, and then write a function which can handle any number of lines.

(2) Hill Cipher: Write functions in Python to encrypt and decrypt text messages using the Hill Cipher. In this project you will also learn about fun matrix operations and how to use NumPy! First, manually code the case of a $2 \times 2$ matrix encryption and decryption, and then use NumPy to write a function for a matrix of size $n \times n$.

## 2. Classical Cryptanalysis

(3) Cracking the Vigenere Cipher: In this project, you will learn how to crack the Vigenere Cipher! You will learn about the Kasiski examination method and Index of Coincidence to guess the key length, and then use frequency analysis Chi-Squared scoring to then determine the key. A nice introduction can be found at `https://macs4200.org/chapters/06/4/chi-squared-scoring.html` and chapter 8 of the following resource: `https://macs4200.org/chapters/08/1/identifying-polyalphabetic.html`.

(4) Autokey Cipher and Cracking the Autokey Cipher: Write a function in Python to encrypt a text message using the Autokey cipher. Then determine a way to decrypt messages, and use Python to help automate the process.

## 3. Modern Encryption

(5) ElGamal: Write functions in Python to encrypt and decrypt integer messages using the ElGamal encryption.

(6) AES: In this project you will learn about the AES encryption, and then use the "pycryptodome" or "cryptography" module to implement the AES encryption. If you have time, try to implement AES in Python without using modules!

## 4. Miscellaneous

(7) Primality Testing: As we have seen, having large prime numbers in very important for modern encryption schemes, but how do we efficiently determine if a number is prime without using a brute-force method? Learn about the Fermat and Miller-Rabin primality tests, and write functions to implement them in Python.
Note: These are probabilistic tests. Efficient deterministic tests are a bit more complicated.