Introduction to Cryptography

Alvin Lin

January 2018 - May 2018

Introduction to Cryptography Project

Implementation Details

This project implements an encryption algorithm based on a family of graphs using Python. The file `cipher.py` contains the library code and the encryption and decryption algorithm. Given a text string and a password, the functions in `cipher.py` will encrypt the text string using the password by turning both into lists of their ASCII code representations and performing the math described in the project specification. The file `gui.py` provides a user friendly GUI interface to the cipher, allowing for a file to be selected and encrypted.

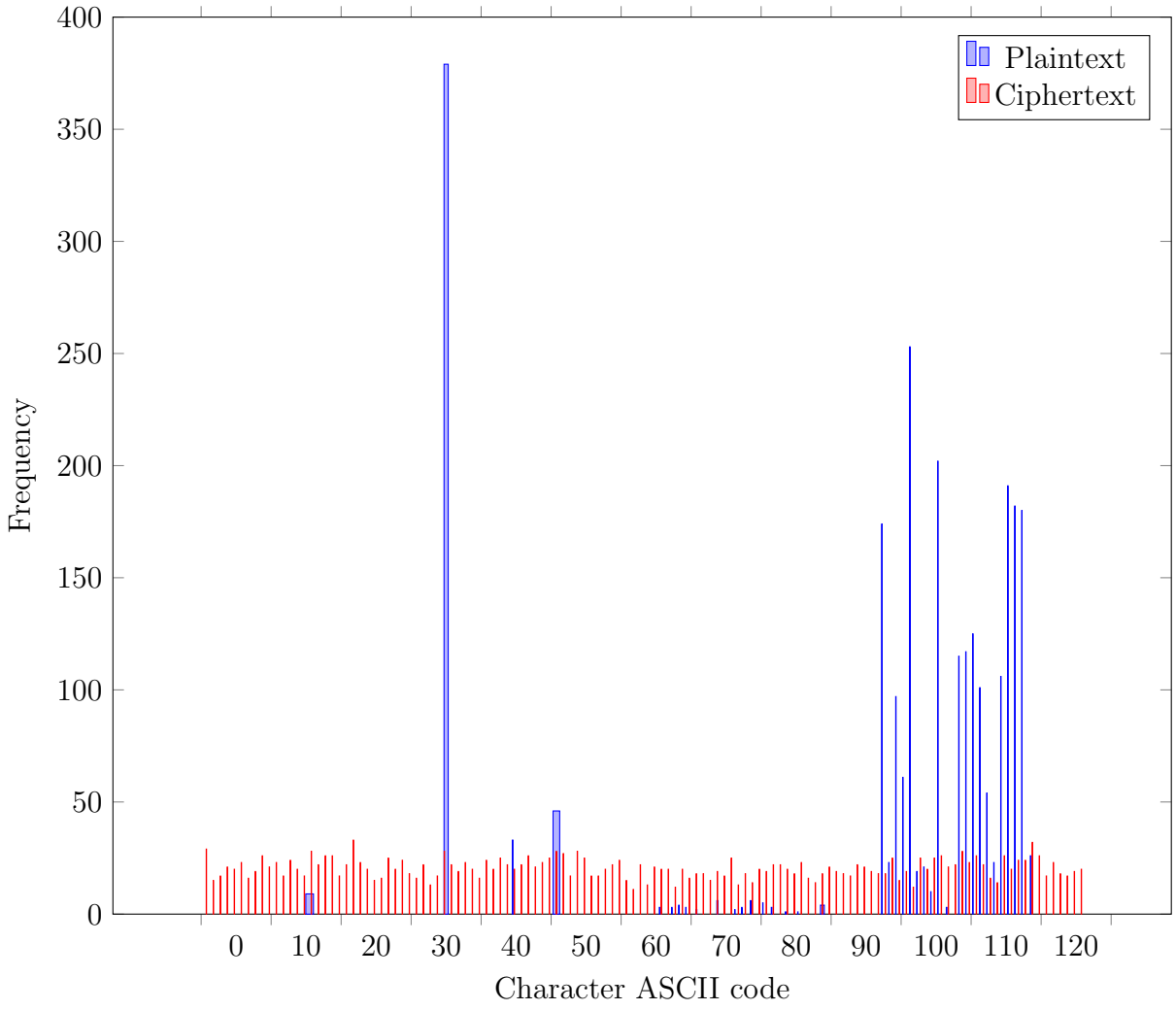
Since we are using a prime modulus of 127 in the algorithm, the character range that the cipher can handle is limited the ASCII decimal values 0 to 126. To ensure that all file types can be encrypted by our algorithm, we first run the file data through base64 to limit the input character set to alphanumerics, `+`, `/`, and `=`. It is then encrypted and written to the specified file.

Decryption runs the operations in reverse by first decrypting the file and then decoding it using base64. The raw binary data is then written back to the output file.

Analysis

Character Frequency Attacks

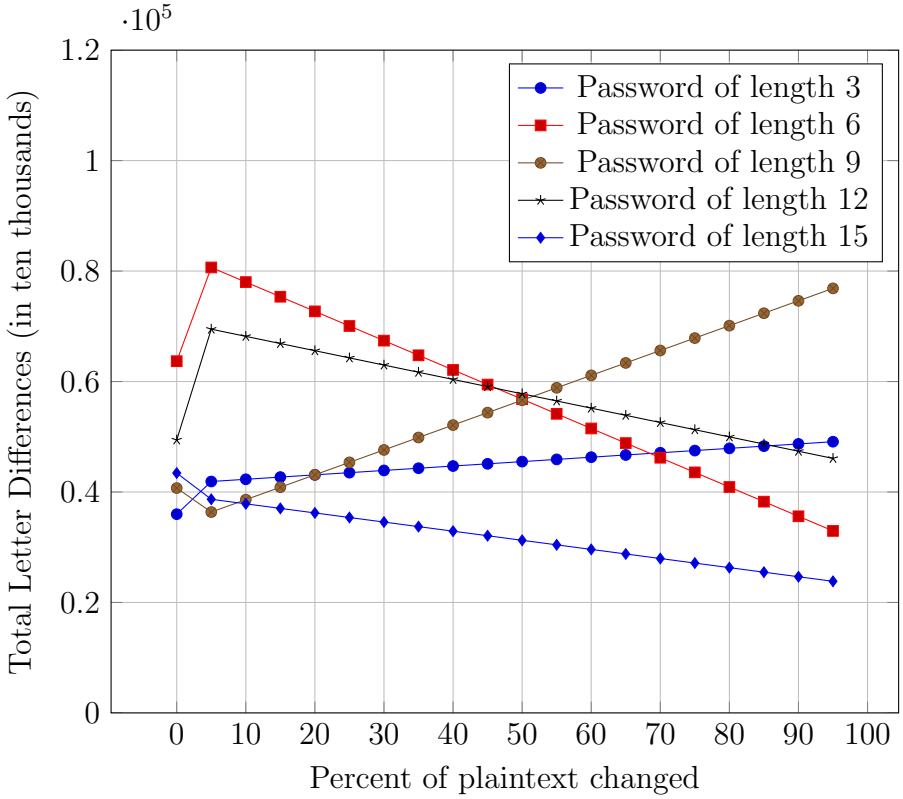
To test if the cipher is resistant to character frequency attacks, a large text file was encrypted and the ciphertext character frequency was compared with the plaintext character frequency. The plot below shows the average character distribution using a long text file with passwords of varying length.



We can see that there is no correlation between the plaintext and ciphertext's character frequencies, making this cipher relative resistant to frequency analysis attacks.

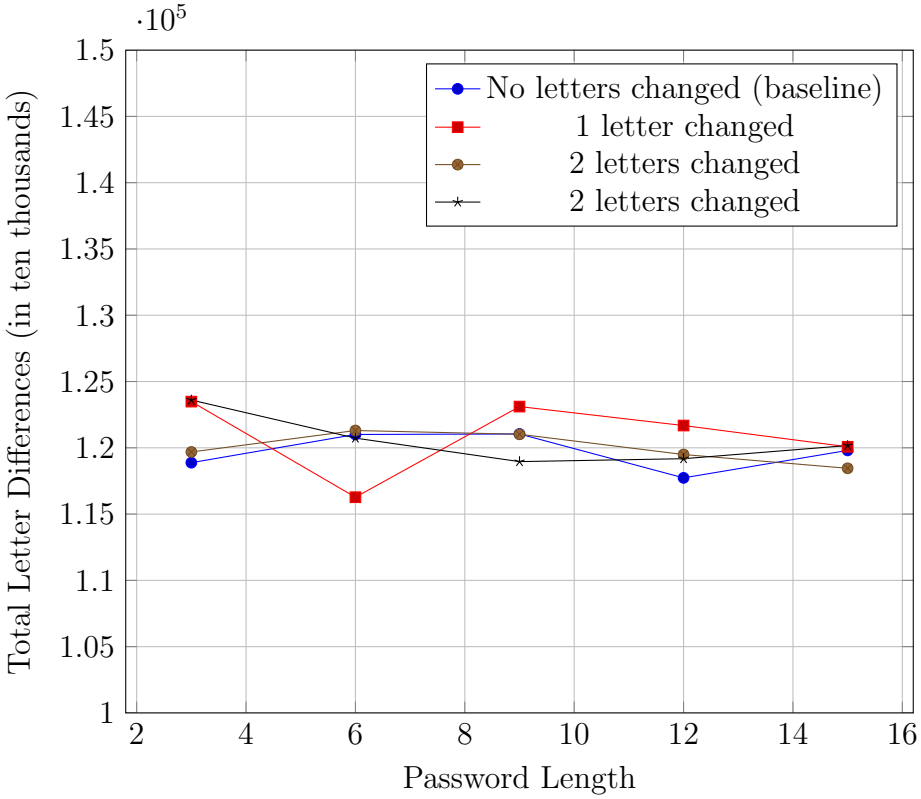
Changes in the Plaintext

Similar plaintexts should yield different ciphertexts even when encrypted with the same passwords. To test if the cipher is resistant to comparison attacks based on this idea, we took a known plaintext-ciphertext pair and replaced a varying segment of the plaintext to compare the shift created in the ciphertext.



Changes in the Password

Like the previous section, similar passwords should also yield different ciphertexts when encrypting the same plaintext. To test if our cipher is resistant to password comparison attacks, we again took a known plaintext but varied the length of the password to see compare the shift created in the ciphertext.


$\cdot 10^5$
1.5
1.45
1.4
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1
Total Letter Differences (in ten thousands)
2
4
6
8
10
12
14
16
Password Length
No letters changed (baseline)
1 letter changed
2 letters changed
2 letters changed