The Cryptanalysis of Homophonic Substitution Ciphers using Mean Squared Error of Letter Frequency

Introduction:

Team members for this project are Kelly Lunghamer, Miguel Salcedo, and Siddharth Sharma.

Tasks:
Kelly - drafting report, scorer function implementation
Miguel - dictionary attacker implementation, file organization, testing interface
Siddharth - candidate attacker implementation, graphics

High Level Cryptanalysis:

In contrast to simple one-to-one mapping of simple substitution discussed earlier in this course, this project analyzed a homophonic substitution cipher, in which a plaintext letter can be mapped to more than one number in the corresponding ciphertext. Moreover, each number can only be mapped to one letter, so the decryption is unique. When conducting cryptanalysis of simple substitution ciphers, statistically analysis of letter frequency can be used to decrease the exhaustive search on a ciphertext. Looking at the figure below, the letter “e” could be substitution with the highest frequency number as it is the most used letter in the alphabet.

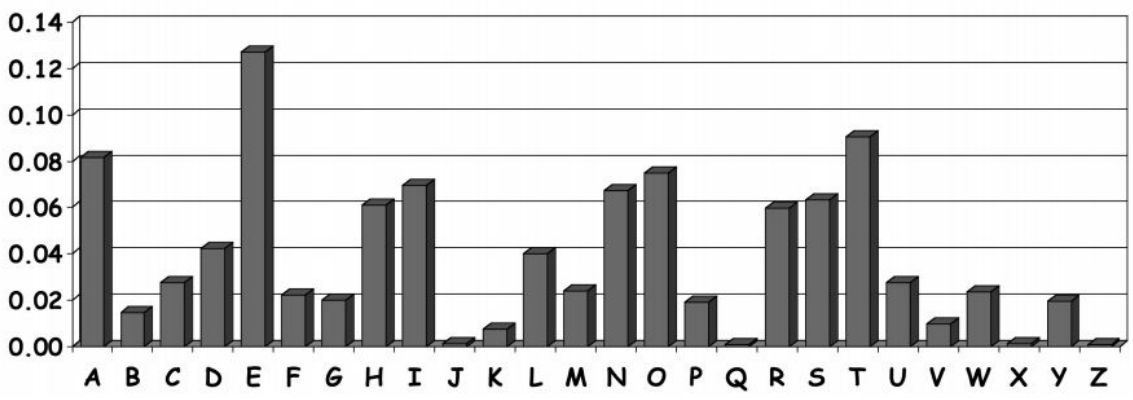


Figure 1: English Letter Frequencies

Nevertheless, using this statistical analysis method is not adequate on a homophonic substitution cipher, so our team brainstormed other methods for cryptanalysis.

The potential exhaustive search method for cryptanalysis would take the dictionary of words and use the ciphertext to build up the plaintext word by word. This method recursively checks and updates the key with the growing plaintext. By using only this implementation, the recursive search would eventually find a correct key, but takes too much time to be effective for decryption.

In this case where the key space is too large to decrypt the ciphertext with exhaustive search, hill climbing is another cryptanalysis technique that uses heuristics to find a reasonable solution give a time constraint. Heuristics use educated guesses to improve upon the current solution using some sort of scoring evaluation. Although heuristics are not guaranteed to lead to an optimal solution, they are generally quicker than brute force solutions.

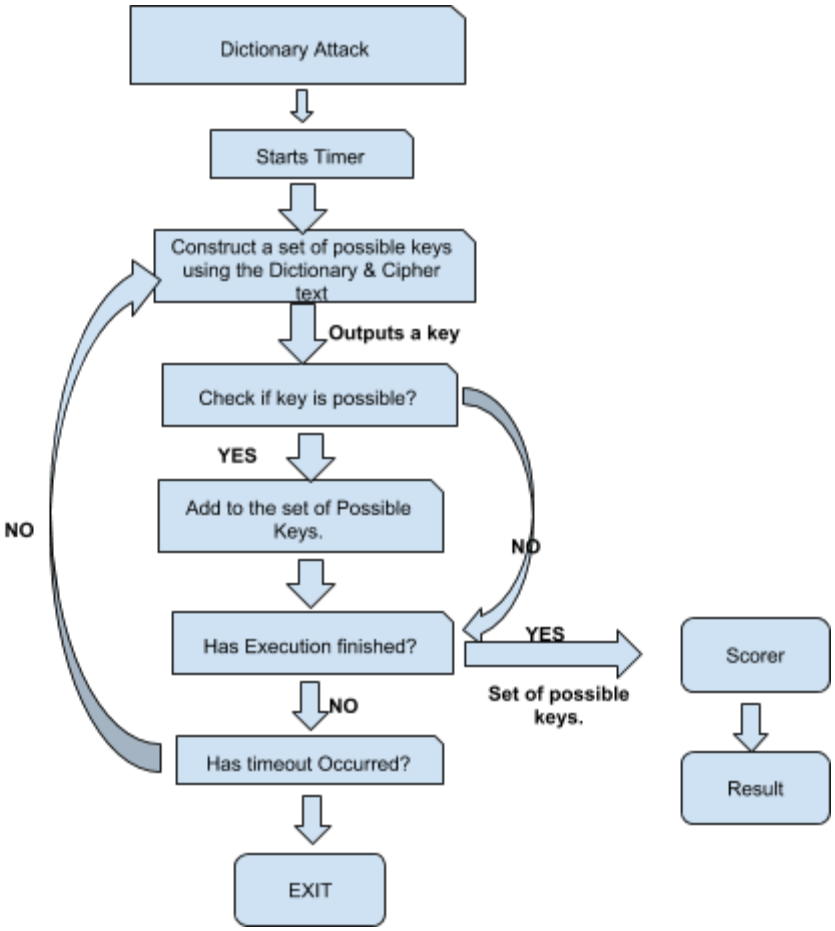
Our final solution uses a hybrid approach between recursive dictionary search and hill climbing. This decryption method is sufficient for the second test, which also makes it sufficient for the first test. First, the encrypted dictionary of words are used as a sample plaintext. As the decryption function builds upon the plaintext and looks for potential keys, the scoring function decrypts the plaintext with each potential key, counts the frequencies of the letters. Then, it determines which key (and hence cipher) has the lowest mean squared error compared to our given letter frequency. Mean squared error is calculated by: subtracting the actual value from expected frequency value for each letter, squaring that error, summing all the errors, and then finding the mean. The cipher with the lowest mean squared error was used to successfully decrypt the ciphertext.

Decryption Algorithm Explained:

The below flow chart explains the working of our dictionary attack. Following are the steps in our dictionary attack:

1. Firstly, we start a timer to control the execution period of our program so that it does not run infinitely.
2. Then we start by constructing the set of possible keys by using the dictionary and the ciphertext is given to us.
3. Once a possible key has been created, we check for its validity using the frequency rules given in the problem statement. If the key is valid we add it to our set of possible keys.

4. After getting a possible key, programs checks whether we have finished the execution or not. If not then it checks for whether a timeout has occurred or not. If the timeout has occurred we simple exit the program, else we go again to step 2 of constructing the set of keys.
5. If execution has finished, we feed the set of possible keys to the Scorer module of our program, which based on the mean square error gives us the best possible key as the output.



Extra Credit (topic #3):

While our cryptanalysis strategy appeared to be successful, we did notice a great increase in runtime as the number of words increased. When the decryption ran using a dictionary of 40 words, the plaintext was outputted within 1-2 minutes, but once the input of words increased to 50, the runtime increased dramatically.

Sources:

- http://www.cs.sjsu.edu/~stamp/RUA/homophonic.pdf
- https://www.oreilly.com/library/view/machine-learning-with/9781785889936/669125cc-ce5c-4507-a28e-065ebfda8f86.xhtml