

7402 Assignment 2

Report/Testing

Aing Ragunathan A00765949

Report

The experiment brought on predictable results since I was using pre-made functions from the past lab to decrypt the ciphertexts. The encrypted ciphertext did look like the text had a decent amount of diffusion and confusion with the naked eye, although a simple analysis of character frequency showed the exact same pattern as a ciphertext encrypted with a Caesar cipher. These similar characteristics drastically reduces my initial assumption that there was indeed a sufficient amount of diffusion and confusion in the transposition's ciphertext. Even though the transposition cipher's results were not as great as I hoped, I believe it could be made much more powerful with some modifications.

This transposition cipher would become increasingly difficult to break if we were to add additional layers of encryption to it. For example using a multi transposition cipher, running the ciphertext through the transposition cipher again, would make it far more difficult to reverse and identify dictionary words. Another example would be to use a columnar transposition cipher to add confusion to the original key. Furthermore, adding completely different ciphers like a Caesar cipher to the transposition ciphertext wouldn't improve diffusion, but it would slightly improve confusion since the a second key is being used. Multiple iterations of the transposition cipher and adding additional ciphers like the Caesar cipher would also increase the confusion of the ciphertext since it forces an analyst to find multiple keys with multiple layers of ciphertexts.

In conclusion, even though the transposition based ciphertexts were easy to break by simply trying different size keys and making look ups in a dictionary, it could be used in conjunction with other techniques such as using as large of a key size as possible, iterating the ciphertext though the cipher multiple times, and using it with other ciphers to increase confusion. These techniques however, do not improve diffusion since the a simple analysis of the character frequency chart shows that the iconic shape of english language only shifts. With this in mind, I believe the best way to improve this cipher is to manipulate characters using a pattern that flattens the character frequency across as large of a range as possible.

Usage:

To brute force a cipher-text string from command line argument:

```
$ python a2.py -s <input string> -k <max key to try>
```

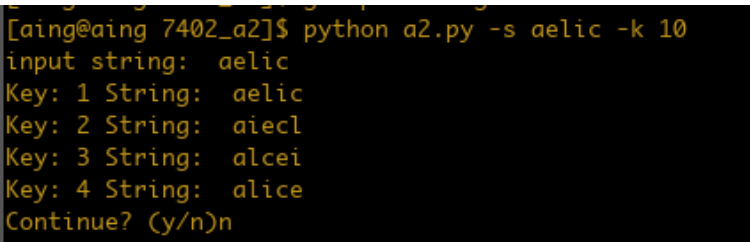
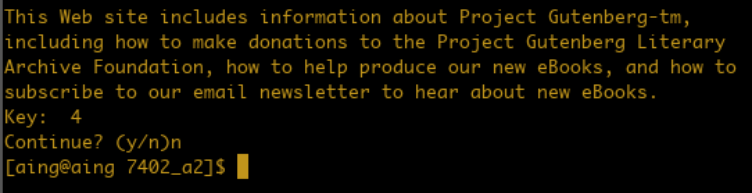
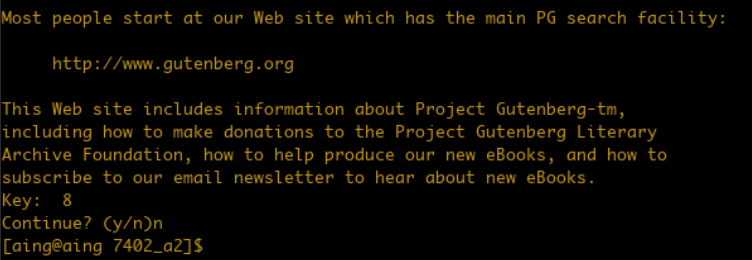
e.g. python a2.py -s aelic -k 4

To brute force a cipher-text from a file:

```
$ python a2.py -i <input file> -k <max key to try>
```

e.g. python a2.py -i alice_encoded.txt -k 10

Testing:

Task	Input	Output
Brute force a simple string like “aelic” with a maximum key size of 10. The result should be “alice” with a key size of 4	\$ python a2.py -s aelic -k 10	
Brute force an entire encrypted book “Alice’s Adventures in Wonderland” with a maximum key size of 10. The result should be the plaintext of the book with a key size of 4	\$ python a2.py -i alice.txt -k 10	
Brute force an entire encrypted book “Moby Dick” with a maximum key size of 10. The result should be the plaintext of the book with a key size of 8	\$ python a2.py -i moby_encoded_8.txt -k 10	

Attempt to brute force a simple string like “aelic” with a maximum key size of 1. The program should not be able to give any solutions	\$ python a2.py -s aelic -k 1	<pre>[aing@aing 7402_a2]\$ python a2.py -s aelic -k 1 input string: aelic Key: 1 String: aelic [aing@aing 7402_a2]\$</pre>
Attempt to brute force an entire encrypted book “Alice’s Adventures in Wonderland” with a maximum key size of 1. The program should not be able to give any solutions	\$ python a2.py -i alice_encoded.txt -k 1	<pre>0teteapacovtatsadeptanota s tennar taintswsadee ca mtra me eweagaiaciloi a5ts tSe pnrie f tscia o hpoam s tdepth umsW revoevwtcit clc S ANrtitsufma tlesesh:g. iwaTd i tunrseh h ianqetwnoooinatciultdtsoos ha rh hf dt etaoi tlap,t nm tncenarm dtsceroitUetsU sosprasfPsh j erea r antsdds noaap eo.eanmob j ermeo k foie.rseia tPeGng c aby cnwsao flhdtneFtte, ddsbdottetBstnaonoooot p.PeGng oaone eapgnscse se er po clciapiaa t.M pst eihm scai t/wteo sbtdndiriatottetiugw enothrcub eyceut,w poerwo,dw sboranleoab o [aing@aing 7402_a2]\$</pre>
Brute force an entire encrypted book “Alice’s Adventures in Wonderland” with a maximum key size of 100. The result should be the plaintext of the book with a key size of 50	\$ python a2.py -i alice_encoded_50.txt -k 100	<pre>http://www.gutut Project GutenbWeb site includes information aboations to the Proeng-tm, including how to make donoundation, how toject Gutenberg Literary Archive F how to subscribe help produce our new eBooks, andabout new eBooks Key: 50 Continue? (y/n)n [aing@aing 7402_a2]\$</pre>