

Section 1:

When choosing a viable corpus to reproduce this on your own, you should select a large, diverse body of text that reflects typical language usage. You could use popular books such as the Lord of the Rings novels. You can use a similar corpus like my own if you are planning to replicate it [here!](#)

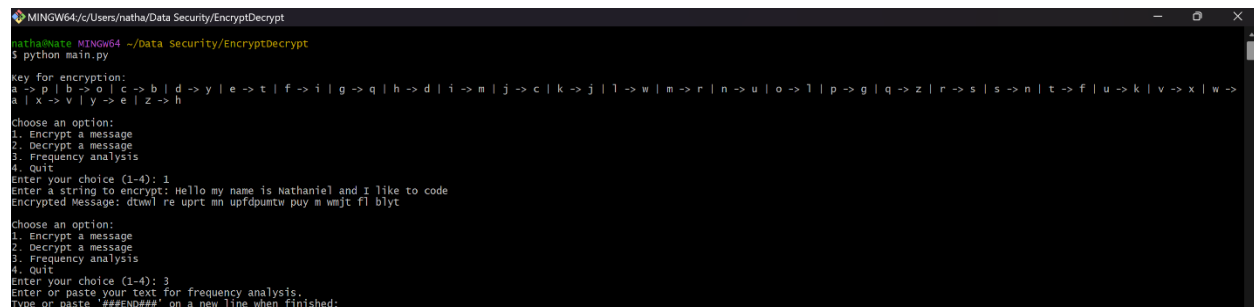
Section 2:

The program is a substitution cipher implementation that allows users to encrypt and decrypt messages using a randomly generated key. I create a key mapping of the alphabet to a shuffled version of itself, the encryption process, showing its mapping to reverse key decryption. Users can also perform frequency analysis on encrypted text to determine the relative frequency of letters, which can help in deducing the original letter mappings.

Section 3(How To):

1. Open Your Terminal. On a Windows computer: Click the Start menu, type "Command Prompt" or "PowerShell," and press Enter. On a Mac: Open the Applications folder, then go to Utilities, and double-click on "Terminal."
2. Go to the Right Folder: You need to tell the terminal where your program is. Type `cd "path_to_your_program"` and hit Enter. Replace "path_to_your_program" with the name of the folder where you saved the program. For example, if your program is in a folder called "ImportantStuff," you would type `cd ImportantStuff`.
3. Run the Program: Type `python main.py` and press Enter. Choose What You Want to Do: Once the program is running, it will ask you what you want to do. You can choose: 1 to encrypt a message (make it secret). 2 to decrypt a message (turn it back into regular words). 3 to do frequency analysis (find out how often letters appear). 4 to quit the program (close it).

Section 4:



```
MINGW64/c/Users/natha/Data Security/EncryptDecrypt
natha@natha MINGW64 ~/Data Security/EncryptDecrypt
$ python main.py
key for encryption:
a->p | b->o | c->b | d->y | e->t | f->i | g->q | h->d | i->m | j->c | k->j | l->w | m->r | n->u | o->l | p->g | q->z | r->s | s->n | t->f | u->k | v->x | w->a | x->v | y->e | z->h
choose an option:
1. Encrypt a message
2. Decrypt a message
3. Frequency analysis
4. Quit
Enter your choice (1-4): 1
Enter a string to encrypt: Hello my name is Nathaniel and I like to code
Encrypted Message: dtwll re uprt mn upfdumtw puy m mmjt fl blyt
choose an option:
1. Encrypt a message
2. Decrypt a message
3. Frequency analysis
4. Quit
Enter your choice (1-4): 3
Enter or paste your text for frequency analysis.
type or paste '#####' on a new line when finished:
```

In this first screenshot we can see the first three requirements. Key-generation (Alice's view), Encryption (Alice's view), and Decryption (Bob's view).

The key generation is shown first matching each letter of the alphabet randomly to another letter of the alphabet with no duplicates.

Following the key generation, I selected option 1(Encrypt a message). In this Encryption phase Alice enters a plaintext message that she wants to encrypt using the key generated earlier. The encrypt function substitutes each letter in the plaintext according to the generated key resulting in the output being ciphertext. I wanted to encrypt “Hello my name is Nathaniel and I like to code” and my ciphertext ended up being “dtwwl re uprt mn upfdpumtw puy m wmjt fl blyt” according to my key generation.

Next, we can see the decryption in action, Bob’s view. Bob, on the receiving end, decrypts the ciphertext using the key by mapping each letter in the ciphertext back to its original plaintext letter. We can see that Bob turned this message: “dtwwl re uprt mn upfdpumtw puy m wmjt fl blyt” back into our original plaintext: “Hello my name is Nathaniel and I like to code”

```

MINGW64~/c/Users/nath/Data Security/EncryptDecrypt
Toronto, ON, M4W 1A8, Canada
http://www.harpercollinsebooks.ca
New Zealand
HarperCollinsPublishers (New Zealand) Limited
P.O. Box 1
Auckland, New Zealand
http://www.harpercollinsebooks.co.nz
United Kingdom
HarperCollins Publishers Ltd.
77-85 Fulham Palace Road
London, W6 8JB, UK
http://www.harpercollinsebooks.co.uk
United States
HarperCollins Publishers Inc.
10 East 53rd Street
New York, NY 10022
http://www.harpercollinsebooks.com
#####
Letter Frequencies in the encrypted text (%):
t -> 12.39%
f -> 0.03%
p -> 8.15%
l -> 7.89%
u -> 6.91%
d -> 6.57%
m -> 6.42%
s -> 5.94%
n -> 5.94%
y -> 5.20%
w -> 4.49%
a -> 2.69%
k -> 2.56%
i -> 2.53%
q -> 2.44%
r -> 2.27%
e -> 1.92%
o -> 1.72%
b -> 1.63%
g -> 1.35%
j -> 0.90%
x -> 0.84%
v -> 0.07%
c -> 0.06%
z -> 0.06%
h -> 0.04%

Potential key pairings based on frequency analysis:
t -> e | f -> t | p -> a | l -> o | u -> i | d -> n | m -> s | s -> h | n -> r |
y -> d | w -> l | a -> c | k -> u | i -> m | q -> w | r -> f | e -> g | o -> y |
b -> p | g -> b | j -> v | x -> k | v -> j | c -> x | z -> q | h -> z

choose an option:
1. Encrypt a message
2. Decrypt a message
3. Frequency analysis
4. Quit
Enter your choice (1-4):
Invalid choice. Please enter a number between 1 and 4.

choose an option:
1. Encrypt a message
2. Decrypt a message
3. Frequency analysis
4. Quit
Enter your choice (1-4): 4
Goodbye! :D

```

Taking a look at Eve’s view with a large body of credible text as our example, we see Eve attempting to break the cipher by using frequency analysis on the ciphertext. Eve is comparing the letter frequencies in the intercepted ciphertext to standard English letter frequencies to potentially deduce the original mapping. We can see that based on the percentages provided ex. t being 12% and based on Relative frequency in the English language chart you provided as well as charts found online e is used 12% in the English language. Thus, Eve can conclude t (cipher text) corresponds to e (plaintext). Eve does this for the first few letters continuing to match other frequent letters in the ciphertext, such as 'r', 'l', and 'n', with their closest counterparts in the English language based on the frequency charts.