Homework 1 - Information Security

Catherine Baker

February 2024

1 Substitution Cipher

1.1 Frequency Table

Letter	Frequency
a	5
b	1
c	4
d	4
e	22
f	31
g	27
h	12
i	10
k	29
l	24
m	1
n	6
o	31
q	28
r	14
s	14
t	39
u	8
v	5
w	7
x	9
y	6
z	34

1.2 Decryption

The decrypted text reads:

Blockchain is a decentralized and distributed digital ledger
technology that enables secure, transparent, and tamper resistant
record keeping of transactions across a network of computers It
serves as the underlying technology for various cryptocurrencies
with Bitcoin being the first and most wellknown application
However, the potential applications of blockchain extend far
beyond cryptocurrencies impacting various industries

To find this I used frequency analysis by creating a simple python program cipherBreak.py that substituted letters in a given text based on a manually updatable dictionary. I first replaced the letter 't' with 'e' in the text based on the frequency analysis tables. I then continued updating the replacement dictionary based on frequency analysis of individual letters (like 'q' -¿ 'a') as well as two and three letter words (like 'and' and 'the') in the text. As these were filled in I expanded to 4 and 5 letter words (like 'that' and 'these'), and on until I had an entire visible message. The final set of replacement letters can be found in my included code (for every [key: value] pair, the key is the substitution cipher letter, and the value is the decoded letter), but is included below for the reader's benefit:

replacements = {'t': 'e', 'q': 'a', 'f': 'n', 'r': 'd', 'z': 't', 'i': 'h', 'l': 's', 'o': 'i', 'v': 'w', 'g': 'o', 'k': 'r', 'a': 'k', 'h': 'p', 'o': 'i', 'c': 'v', 'u': 'g', 'w': 'b', 'y': 'f', 'i': 'h', 's': 'l', 'n': 'y', 'W': 'B', 'O': 'I', 'e': 'c', 'm':'z', 'x': 'u', 'd':'m', 'I': 'H', 'b':'x'}

2 Encryption with Caesar and Affine Ciphers

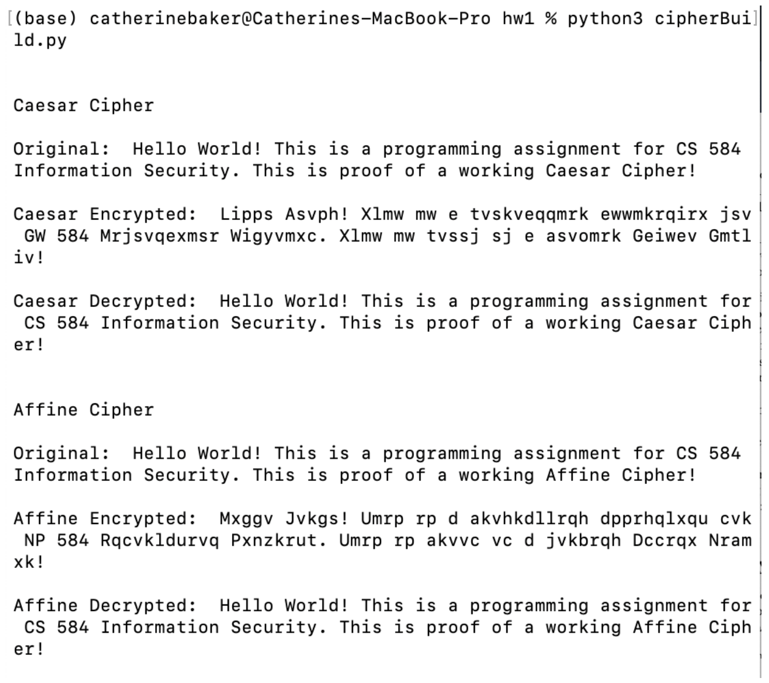
My code file cipherBuild.py can encrypt and decrypt in Caesar and Affine Ciphers. Example usage is given in the __main__ section of the file. An example output with a Caesar key of 4, Affine a = 5 and b = 3 is given in Figure 1.

Additionally my program will print if the Affine chosen a has no inverse with the given mod 26. Here is another example with Caesar key = 15 and Affine a = 10 (which has no modular inverse with 26) and b = 4 in Figure 2.

My code includes no external libraries. Further testing of my code can be done by adding lines to the __main__ section in the following format, for the Caesar Cipher:

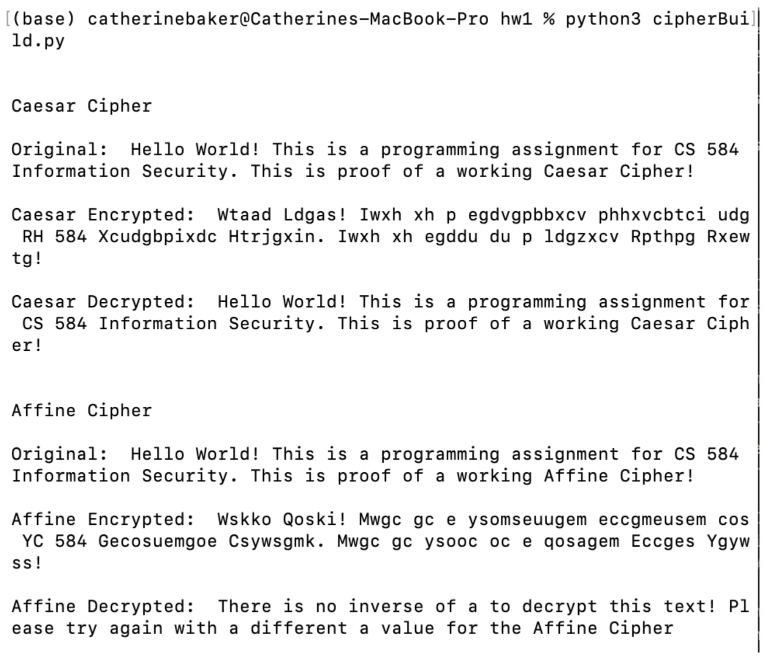
```
# Test Caesar Cipher
cKey = {shift value}
cText = "{Text to encode}"
print("\nOriginal: ", cText) # print original text
cResult = caesarEncrypt(cText, cKey)
print("\nCaesar Encrypted: ", cResult) # print encrypted text
cOriginal = caesarDecrypt(cResult, cKey)
print("\nCaesar Decrypted: ", cOriginal) # print decrypted text
```

and for the Affine Cipher:



```
[(base) catherinebaker@Catherines-MacBook-Pro hw1 % python3 cipherBui]
ld.py


Caesar Cipher

Original:  Hello World! This is a programming assignment for CS 584
Information Security. This is proof of a working Caesar Cipher!

Caesar Encrypted:  Lipps Asvph! Xlmw mw e tvskveqqmrk ewwmkrqirx jsv
 GW 584 Mrjsvqexmsr Wigyvmxc. Xlmw mw tvssj sj e asvomrk Geiwev Gmtl
iv!

Caesar Decrypted:  Hello World! This is a programming assignment for
 CS 584 Information Security. This is proof of a working Caesar Ciph
er!


Affine Cipher

Original:  Hello World! This is a programming assignment for CS 584
Information Security. This is proof of a working Affine Cipher!

Affine Encrypted:  Mxggv Jvkgs! Umrp rp d akvhkdllrqh dpprhqlxqu cvk
 NP 584 Rqcvkldurvq Pxnzkrut. Umrp rp akvvc vc d jvkbrqh Dccrqx Nram
xk!

Affine Decrypted:  Hello World! This is a programming assignment for
 CS 584 Information Security. This is proof of a working Affine Ciph
er!
```

Figure 1: "Caesar Encrypted" text is the product of the Caesar encryption of the Original text through my Caesar encryption algorithm. "Caesar Decrypted" text is this encrypted text run through my decryption algorithm in cipherBuild.py. These definitions are parallell for the Affine Cipher.



```
[(base) catherinebaker@Catherines-MacBook-Pro hw1 % python3 cipherBui]
ld.py


Caesar Cipher

Original:  Hello World! This is a programming assignment for CS 584
Information Security. This is proof of a working Caesar Cipher!

Caesar Encrypted:  Wtaad Ldgas! Iwxh xh p egdvgpbbxcv phhxvcbtci udg
 RH 584 Xcudgbpixdc Htrjgxin. Iwxh xh egddu du p ldgzxcv Rpthpg Rxew
tg!

Caesar Decrypted:  Hello World! This is a programming assignment for
 CS 584 Information Security. This is proof of a working Caesar Ciph
er!


Affine Cipher

Original:  Hello World! This is a programming assignment for CS 584
Information Security. This is proof of a working Affine Cipher!

Affine Encrypted:  Wskko Qoski! Mwgc gc e ysomseuugem eccgmeusem cos
 YC 584 Gecosuemgoe Csywsgmk. Mwgc gc ysooc oc e qosagem Eccges Ygyw
ss!

Affine Decrypted:  There is no inverse of a to decrypt this text! Pl
ease try again with a different a value for the Affine Cipher
```

Figure 2: The plot produced by my CatherineBakerQ1B.py code.

```
# Test Affine Cipher
a, b = {aValue}, {bValue}
aText = "{Text to encode}"
print("\nOriginal: ", aText) # print original
aResult = affineEncrypt(aText, a, b)
print("\nAffine Encrypted: ", aResult) # print encrypted text
aOriginal = affineDecrypt(aResult, a, b)
print("\nAffine Decrypted: ", aOriginal) # print decrypted text
```