

Cryptography & Data Privacy (CSCI 4525) – Assignment 1

Sami Cemek
ahmet.cemek24@ncf.edu

Instructor: Daniel Page

Date Completed: 02/21/24

I'm using Python for this assignment.

Question 1:

- a) Code submitted on Canvas.
- b) My code does Caesar Cipher Decryption with an exhaustive-key search. We don't know the shift (k). We just have the secret message shifted by k. I provide an option of putting the message as an input or just changing the `secret_message` variable. I have a loop that iterates through each possible shift value from 0 to 25. Since there are 26 letters in the English alphabet, Caesar Cipher has 26 possible shift values. `ord(char)` returns the ASCII value of the character. $(\text{ord}(\text{char}) + \text{shift} - 65) \% 26 + 65$ shifts the character back by shift positions. The $- 65$ normalizes the ASCII values so that 'A' has the value of 0, and the $\% 26$ ensures that the result wraps around if it goes beyond 'Z'. Finally, $+ 65$ brings it back to the ASCII range for uppercase letters.

The program output is this:

```
a Privacy/Assignments/Crypto_Assignment1/assignment1.py"
The decrypted message with shift 0 is: ZWOPFLQFEEFKONFIQBFQEZDGFIFKREKGQIFSCVDJQQZKQJZQEFKQCBVCPQYRKQPFQLNZCZCQFQZDFGFIKREKQNFBIQZIYTIRUQYRDDZEX
The decrypted message with shift 1 is: AXROGMVRGFGLROGTCRGFAEHGJLSFLRHJGTDWEKRRAILAKRFLRDAHCQRLZSLRQGMROADDRVGRAEHGJLSFLROGJCRRJAUZSVRZSEEAFY
The decrypted message with shift 2 is: BYSRINSMHGMSPHIKDSHGSBFIIKMTGMSIKHUEKFSLSBMBSLGMSSEBDXERSMATMSRHNHSPEEESNHBSFBIKMTGMSPHDKSSBVATKWSATFBGZ
The decrypted message with shift 3 is: CZTSIOTXIHINTQILETIHTCGJILNUHNTJLIVFYGMITCTNTHINTCEYESTBUNTSIOTQCFXTXTCGJILNUHNTQIETTLLWBULXTBUGGCHA
The decrypted message with shift 4 is: DAUTJPUJIIJOURJMFIJUDHKJMOVIOKUMWJGZHNUUDQDNUIJJOUJDFZGTUOCVOUTJPURDGUYJUDHKJMOVIOURJMFUUMDXCVMYUCVHHDIB
The decrypted message with shift 5 is: EBVKUQVKZKJPKPSKNGVKJVEILKNPWPJPNLXOXAIIOWEPVEOVJJKPVHEGAHUVPHDPMVPUKVQSEHNVZKEVILKNPWPJPSKNGVNEYDWNZDWIIIE3C
The decrypted message with shift 6 is: FCWLRWALKLQTLQHMLKWFJMLQXQWQJOLYIBJPMWFQWPMKLOJWTFBILWQEXQJWLRTFJLALWFJMLQXQWQJLWHMWFZEXOAWEJXJFD
The decrypted message with shift 7 is: GDXWMSXBMIMRXUIMPXTMLXGKNNPRLRNXPMPZJCKQXRGRGXQXLMRXJGCIJWXRFYRXXMSXUGJTXBMXGKNNPRYLRXUIMPDXPAFYPBXFYKKGLF
The decrypted message with shift 8 is: HEYXNTYCNMNSVNQJYNNHYHLNQSNQZMSYQNAKDLRYHYSYHRYMNSYKHDIXYSGZSYNTVYHKKYCNVHLNQSZMSYVNQJYYQHBGZQCYGZLLHMF
The decrypted message with shift 9 is: JGAZPVAEPOPUXPLAOPJNPSUBQUAQSPCMFTAAJUATAOUMLFMZAUTBUAPZPVAXJMAEPAJNQPSUBQAUXPSSLASJDIBSEATBNJNJOH
The decrypted message with shift 10 is: KIBAQMBFPOQWBYQTMQBPBKQRQTVCPVBRQTDQNGUBBKVBKUPQVBNKMGNAVBJVCAQWBYKNNFBQKQRQTVCPVBYQTMBTKEJCTFBJCOOKPI
The decrypted message with shift 11 is: LICBRXGQRWCZRUNCQCLPSRUWQNCSEUREQHPCVNLCLVCORWCOLNHBKWDMBRCXZLQOCGRCLPSRUWQNCZRUNCULFKDUGCKDPLQJ
The decrypted message with shift 12 is: MDCSYDHRSRSXASVDSRDMQTSVXERXDTVSFPJOMDDMDMUDRSXPDMOIPCDXLXEDCSYDAMPPDHSDMOTSVXERXASVODDMVGELEVHLDLEQMRK
The decrypted message with shift 13 is: NKEDEZEITSTYEBTWPETSENRTUFYFSYEWTGQJRXEEENYENKESTYEQNPJQDEYMFYEDTEZBNQQEETENRUTWYFSEYBTWPPEENNIMFWIEMFRNISL
The decrypted message with shift 14 is: OLEUFAQJUTZCUXQFUTQFOSVUXZGTZFXVUHRKSYFFOFTUZFRQKZFEUAFCORRFJUFSOSVUXZGTZFCUXQFFXOINGQJFNGSSOTM
The decrypted message with shift 15 is: PMGFVGPKVUAVGDVYRGVUGPTWYAHUAGWVYISLTZGGPAGZGUAGSPRLSGFAOHAGFVBGDPSSGKVGPTWYAHUAGDVYRGYPJOHYKGHTTPUN
The decrypted message with shift 16 is: QNHGMCHLWMWBHEWZSHWVHQXWZBIVBIXZWTJTMIAHHQBHQAHWMBHTQSMTGHIBPJBHGMCHEQTTHLWHLQJXIZBTVBHEWZSHZQKPTLHPTIQUVO
The decrypted message with shift 17 is: ROIHXDMWXCIFXATDXWIRVYXACJWCYAKUNVBIIRCRIBXCIURTNHUQJCIHXDIFRUUIMXRVYXACJWCIXATIARLQJAMIQJVRWP
The decrypted message with shift 18 is: SPJIEJNYXDJDGYBUJYXSWSZYBDKQDZBYLVONCJJSQJXYDJVSUOVIDRQDJIYEGSVVJNYJSWZBIDKQDGYBUJBSMRKBENJRKWMSXQ
The decrypted message with shift 20 is: TQKJZFKOZYZEKHZVKCZKTXAZCELYEKAZMIPDXKCTKTDKYZEKWTVPWJKESLEKJZFKHTWKOZKTXAZCELYEKAHZVKKCTNSLCKSLXXTYR
The decrypted message with shift 21 is: URLKAGLPAZAFLIADLZLUYBADMZFLBDANXQYLLFLUUELZAFLXUQXKLFTMFLKAGLJUXLPALUYBADFMZFLIADNLLDUOTMDPLTMYYUZS
The decrypted message with shift 22 is: VSMBLHQBABGMUBXEMBAVMZCBEGNAGMCEBOYRFMMVGMFMBGMYYXRYLQUNGMLBHMDVYVYQBMVZCBEGNAGMUBEXMEVPUNEQMUNZZVAT
The decrypted message with shift 23 is: WTMCMINRCBCHNKCFYNCBNMADCFHOHNDFCPZSAGNMWVHGNBCHNZWYSZMHVHOMCINKWZNRNCMADCFHOBHNCFCYNNFHVVOVFRNVOAABU
The decrypted message with shift 24 is: XUONDJOSDCDITOLDZGZDOCXBEDGIPCIQEDQATBHOOXIOXHODCIAZTANOIPIONDQJOLXAAOSDXBEDGIPCIOLDGZOOGXRPGSQNPBXCV
The decrypted message with shift 25 is: YPOEKPTEDEJPMEHAPEDPYCFCFHJQDJPHERBUCIPPYJPYIPDEJPBYAUBOPJQJPOEKPMBPTEPYCFEHJQDJPMEHAPPYXQJHTPXQCCYD
PS C:\Users\asem\Desktop\Cryptography and Data Privacy\Assignments\Crypto_Assignment1>
```

The decrypted message with shift 9 is:

IFYYOUZDONOTZWORKZONZIMPORTANTZPROBLEMSZZITZISZNOTZLIKELYZTHATZYOUTZWILLZDOZIMPORTANTZWORKZZRICHARDZHAMMING

- c) The key is 9. The letter "Z" is used to indicate escapes between words. Once we clean those, the message is: ***IF YOU DO NOT WORK ON IMPORTANT PROBLEMS IT IS NOT LIKELY THAT YOU WILL DO IMPORTANT WORK RICHARD HAMMING.***

Question 2:

- a) After applying the Euclidean Division Algorithm for $a = 420$ (r_0) and $b = 17$ (r_1), the quotients (q) are and remainders (r) as follows (top to bottom). Basically we divide a by b . This gives us q and r . Our past b becomes our a and our past r becomes our b . We repeat this till r_m is not equal to 0.

i,	a,	b,	q,	r
1,	420,	17,	24,	12
2,	17,	12,	1,	5
3,	12,	5,	2,	2
4,	5,	2,	2,	1
5,	2,	1,	2,	0

$$r_m = \gcd(a, b) = 1$$

- b) Here is the output after writing the code.

```
● PS C:\Users\ascem\Desktop\Cryptography and Data Privacy\Assignments\Crypto_Assignment1>
  a Privacy/Assignments/Crypto_Assignment1/assignment1.py"
a = 420
b = 17
q = 24
r = 12
-----
a = 17
b = 12
q = 1
r = 5
-----
a = 12
b = 5
q = 2
r = 2
-----
a = 5
b = 2
q = 2
r = 1
-----
a = 2
b = 1
q = 2
r = 0
-----
1
○ PS C:\Users\ascem\Desktop\Cryptography and Data Privacy\Assignments\Crypto_Assignment1>
```

- c) We can optimize our algorithm by not storing all the quotients and remainders during each iteration. If we just use a and b until b is not 0. Then we will return a at the end. This approach is more space-efficient as we don't have to store the q and r . Here is my pseudocode.

```
# Pseudocode
# euclidean_division_algorithm_extended(a, b)
#     while b is not 0
#         a, b = b, a % b
#     return a
```

Question 3:

- a) The code is submitted on Canvas. Here is the program's output.

```
PS C:\Users\ascem\Desktop\Cryptography and Data Privacy\Assignments\Crypto_Assignment1> & c:/p/Cryptography and Data Privacy/Assignments/Crypto_Assignment1/assignment1.py"
Elements in Z36 with multiplicative inverses: [1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35]
Total number of multiplicative inverses in Z36: 12
```

- b) Using the formula, we confirm that there is 12 multiplicative inverses in Z_{36} in total.

$$\prod_{i=1}^n (p_i^{e_i} - p_i^{e_i-1})$$

$$\begin{aligned} 36 &= 2^2 \cdot 3^2 \\ \phi(36) &= (2^2 - 2^1) \cdot (3^2 - 3^1) \\ &= (4 - 2) \cdot (9 - 3) \\ &= 2 \cdot 6 \\ &= 12 \end{aligned}$$

Question 4:

For a key k to have the property that for every $x \in Z_{24}$, $e_k(x) = d_k(x)$ it means that for every plaintext symbol, its ciphertext symbol is the same as itself after decryption.

$$\begin{aligned} \text{Encryption: } e_k(x) &= (x+k) \bmod 24 \\ \text{Decryption: } d_k(y) &= (y-k) \bmod 24 \\ e_k(x) &= d_k(y) \\ (x+k) \bmod 24 &\equiv (y-k) \bmod 24 \\ (x+k) &\equiv (x-k) \bmod 24 \\ 2k &\equiv 0 \bmod 24 \\ \rightarrow \text{basically this says } 2k &\text{ divided by 24 should give remainder 0} \\ \text{For } Z_{24} = \{0, 1, 2, \dots, 23\}, k \text{ can be } 0 &\text{ and 12.} \end{aligned}$$

Question 5:

- a) Here are the equations we will be using:

affine cipher $K(3,5)$

$\begin{matrix} 4 & 13 & 2 & 17 & 24 & 15 & 19 \end{matrix}$

encryp

$e_K(x) = (3x + 5) \bmod 26$

$d_K(y) = 3^{-1}(y - 5) \bmod 26$

This is what we get after encryption:

e: $3 \cdot 4 + 5 \equiv 17 \bmod 26 \rightarrow 17 \rightarrow R$

n: $3 \cdot 13 + 5 \equiv 44 \bmod 26 \rightarrow 18 \rightarrow S$

c: $3 \cdot 2 + 5 \equiv 11 \bmod 26 \rightarrow 11 \rightarrow L$

r: $3 \cdot 17 + 5 \equiv 56 \bmod 26 \rightarrow 4 \rightarrow E$

y: $3 \cdot 24 + 5 \equiv 77 \bmod 26 \rightarrow 25 \rightarrow Z$

p: $3 \cdot 15 + 5 \equiv 50 \bmod 26 \rightarrow 24 \rightarrow Y$

t: $3 \cdot 19 + 5 \equiv 62 \bmod 26 \rightarrow 10 \rightarrow K$

- b) This is what we get after decryption:

R: $9 \cdot (17-5) \equiv 108 \bmod 26 \rightarrow 4 \rightarrow E$

S: $9 \cdot (18-5) \equiv 117 \bmod 26 \rightarrow 13 \rightarrow N$

L: $9 \cdot (11-5) \equiv 54 \bmod 26 \rightarrow 2 \rightarrow C$

E: $9 \cdot (4-5) \equiv -9 \bmod 26 \rightarrow 17 \rightarrow R$

Z: $9 \cdot (25-5) \equiv 180 \bmod 26 \rightarrow 24 \rightarrow Y$

Y: $9 \cdot (24-5) \equiv 171 \bmod 26 \rightarrow 15 \rightarrow P$

K: $9 \cdot (10-5) \equiv 45 \bmod 26 \rightarrow 19 \rightarrow T$

Question 6:

- a) Code submitted on Canvas.

- b) My code does Affine Cipher Decryption with Exhaustive-key search. I have helper functions, modinv calculates the modular multiplicative inverse of an integer a modulo m and egcd is the Extended Euclidean Algorithm for finding modular inverse. Each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. The formula used means that each letter encrypts to one other letter, and back again, meaning the cipher is essentially a standard substitution cipher with a rule governing which letter goes to which.

The program output is this:

(Image is not the whole output, I put there to show how it looks)

```

Key: (1, 0), Decrypted text: EILYTAELJDTLHGCJLNQJLUGTTAJLUATJVAFLKJZLEITLUATJVAFLJDILPQJQVILNQJLUGTTAJLYTAELKJZ
Key: (1, 1), Decrypted text: DHXKSZDKICHKGFBKMPKTFSSZIUKSIZKJYIKDHTKZSIUKSICHKOPUHKMPIKTFSZIKXSZDKJY
Key: (1, 2), Decrypted text: CGJWRYCJBGFIEAHJLOHJSERRYHJSYRHTYDJIHXJCGJSYRHTYDJHBGJNOHOTGJLOHJSERRYHJWRYCJIX
Key: (1, 3), Decrypted text: BFTVQXBTAFTEDZGKNGIRDQQXGTRXQGSXCTHGWIIBTRXQGSXCTGAFIMNGNSFIKNGTRDQQXGIVQXRTHGW
Key: (1, 4), Decrypted text: AEHUPWAHFZEHDYCFFHJMFFQCPWFHQWPFRWBHFVVAEHOFPFRWBHFZEHLFMREHJMFFQCPWFHUPWAHFV
Key: (1, 5), Decrypted text: ZDGTOWZGEYDGCBXEGILEGPBOOVEGPVQEAVGFEUGZDGPVQEAVGFEYDGKLEQDGILEGPBOOVEGTOVGFEU
Key: (1, 6), Decrypted text: YCFNSNUYFDXCFCBAWDHFIDFOANNUDFOUNDPUZFEDTFYCFOUNDPUDXFCKDKPCFHKDFOANNUDFSNUYFEDT
Key: (1, 7), Decrypted text: XBERMTXEWBEAZCCEGJCENZMMTCENTMCOTYECWEIJCJOBEGJCENZMMTCERMTXEDCS
● Key: (1, 8), Decrypted text: WADQLSNDBVADZYBUDFTBDMYLLSBDSLBNSDCBBRWDMSLBNSXDBVADHIBINADFIBDMYLLSBDSLNSWDCB
Key: (1, 9), Decrypted text: VZCPKRVCVAUZYCTACEHACLXXKRACLRKAMRWCAQCVZCLRKAMRWCAUZCGHAHMZCEHACLXXKRACPKRVBCAQ
Key: (1, 10), Decrypted text: UYBQJQBZTYBXWSZBDGZBKWJQZBKQJZLQVBZPBUYBKQJZLQVBZTYBFGZGLYBDGZBKWJQZBQJQUBAZP
Key: (1, 11), Decrypted text: TXANIPATYSXAMWRYACYAJVIIPYAJPIYKPUAAYOATXAJPYIYKPUAAYSXAEEFYFKXACFYAJVIIPYANIPATAZO
Key: (1, 12), Decrypted text: SWZMHOSZXWRZVUQXZBEXZIUHHOXZIOHXJOTZYXNZSWZIOHXJOTZXWRZDEXEJWZBEXZIUHHOXZMHOSZYXN
Key: (1, 13), Decrypted text: RYLYGNRYWQVYUTPWYADWYHTGGNMWYHNGWINSYXWMYRVRHNGWINSYWQYCDMDIVYADWYHTGGNWYLGNRXNM
Key: (1, 14), Decrypted text: QUKKFMQXVPUXTSOVXZCVXGSFFMVXGMFVHMRXWVLXQXGMFVHMRXWVUXBCVCHUXZCVXGSFFMVXKMQXWV
Key: (1, 15), Decrypted text: PTWJELPWUOTWSRNUWYBWFREELUWFLEUGLQWVUKWPTWLEUGLQWVUKWPTWLEUGLQWVUKWPTWLEUGLQWVUK
Key: (1, 16), Decrypted text: OSVIDKOVTSVRQMTVXATVEQDDKTVKEDTFKPVUTJVSVEKDTFKPVTSVZATAFSVXATEQDDKTVIDKOVUTJ
Key: (1, 17), Decrypted text: NRUHCJNUSMRUQLSUWSZUDPCCSJSDJCSEJOUTSJUNRDJCSEJOUSMURYZSERUNZSUDPCCSJUHCJNUTS
Key: (1, 18), Decrypted text: MQTBGBIMTRLQTPOKRTVYRTCOBBIRTCIBRDINTSRHTMQTCIBRDINTRLQTXYRDQTVYRTCOBBIRTBGBIMTSRH
Key: (1, 19), Decrypted text: LPSFAHLSQKPSNQSUXQSBNAAHQSBHQCHMSRQGSLSPBHQCHMSQKPSWQXQCPSSUXQSBNAAHQSFALRSQG
Key: (1, 20), Decrypted text: KOREZGKRJPRJORNMIPTWPRAZZGPRAGZPBLRQPFRKORAGZPBLRQJORWPMWPORTWPRAMZZGPREZGKRQPF
Key: (1, 21), Decrypted text: JNQDFYJQOINQMLHQ5VQZLYFOQZFYOAFKQOEQJNQZFYOAFKQOINQVUOANQSVQZLYFOQDFYJQPOE
Key: (1, 22), Decrypted text: IMPCXEIPNHMPLKGPNPURNPYKXXENPYEXNZEJPONDPTMPYEXNZEJPNHIMPNUZMPRUNPYKXXENPCXEIPOND
Key: (1, 23), Decrypted text: HLOBWDHOMGLOKJFMQTMQXJWMDOXDWYDIONMCQHLOXDWYDIONMCQHLOXDWYDIONMCQHLOXDWYDIONMC
Key: (1, 24), Decrypted text: GKNAVCGNLFKNJIELNPSLNWIVVCLNWCVLXCHNMLBNKGNNCVLXCHNLFKNRSLSXKNPNSLNWIVVCLNAVCGNMLB
Key: (1, 25), Decrypted text: FJMZUBFMKEJMIHDKMORKMVHUBKMBUKWBGMLKAMFJMVBUKWBGMKEJMQRKRWJMORKMVHUBKMUZBFMLKA
Key: (3, 0), Decrypted text: KUVIPAKVDBUVLCSDVNODVYCPPADVYAPDHATVMDRVKUVUYPADHATVDBUFVODOHVUNODVYCPPADVIPAKVMDR
Key: (3, 1), Decrypted text: BLMZGRBMUSLMCTJUMEFUMPTGGRUMPRGUYSRKMDUIMBLMPRGUYSRKMSLWUFUFYLMEFUMPTGGRUMZGRBMDU
Key: (3, 2), Decrypted text: SCDQXISDLJCDTAKLDWLGDXXILDGXLPIBDULZDSCDGIXLPIBDLJCDNWLWPCDVWLGDGXILDQXISDULZ

```

After copying the whole output on a text file and looking for a common word like “THE”, I found this:

Key: (7, 6), Decrypted text:

WEXKNOWXTHEXPASTXBUTXCANNOTXCONTROLXITZXWEXCONTROLXTHEFU
TUREXBUTXCANNOTXKNOWXITZ

- c) The key is (7, 6). The letter “X” is used to indicate scapes between words and “Z” is used for period. Once we clean those, the message is: **WE KNOW THE PAST BUT CANNOT CONTROL IT. WE CONTROL THE FUTURE BUT CANNOT KNOW IT.**