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. (ord(char) + 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:

The decrypted message with shift 9 is:
IFZYOUZDONOTZWORKZONZIMPORTANTZPROBLEMSZZITZISZNOTZLIKELYZTHATZYOU ZWILLZDOZIMPORTANTZWORKZZRICHARDZHAMMING

c) The key is 9. The letter "Z" is used to indicate scapes 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 (r0) and b = 17 (r1), 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 pseducode.

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

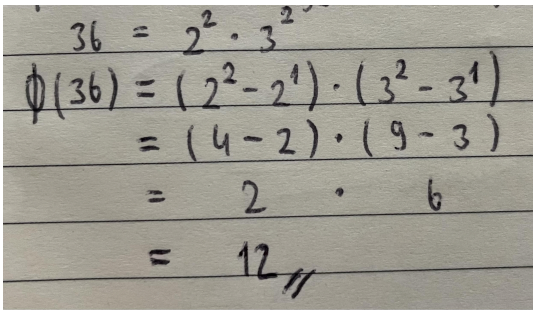
Question 3:

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
```

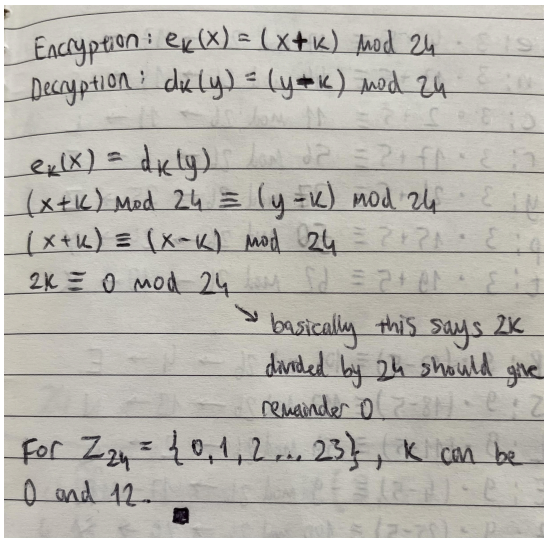
b) Using the formula, we confirm that there is 12 multiplicative inverses in Z_{36} in total.

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



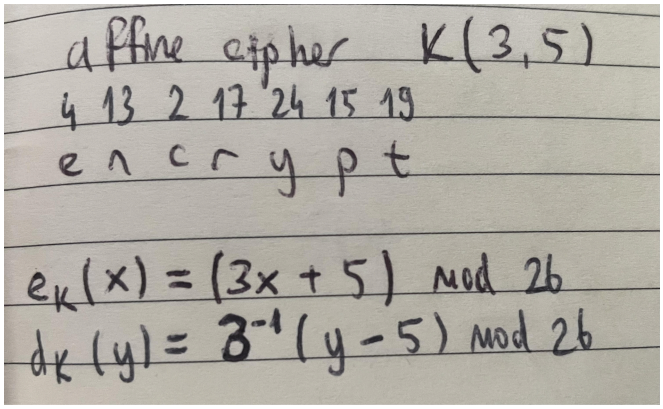
Question 4:

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.

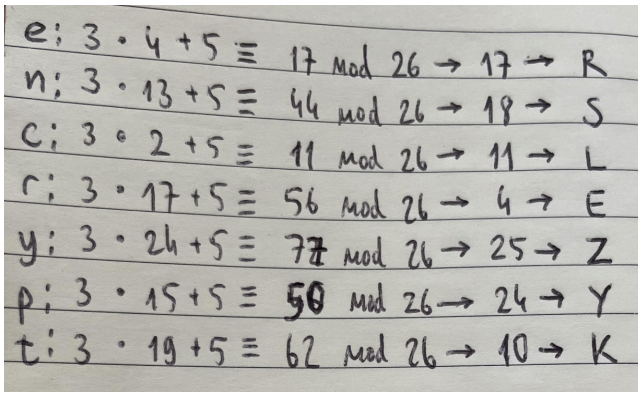


Question 5:

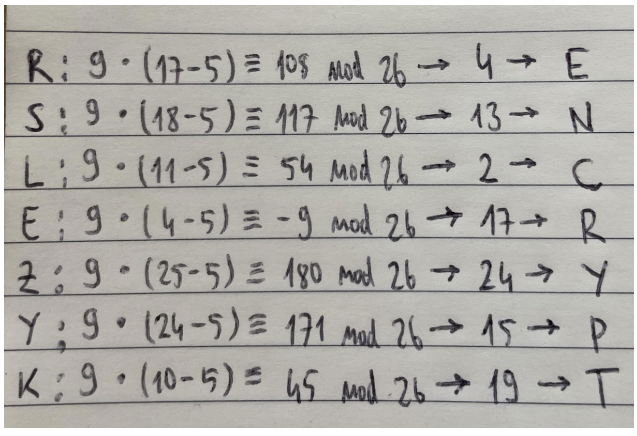
a) Here are the equations we will be using:



This is what we get after encryption:



b) This is what we get after decryption:



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)

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:
WEXKNOWXTHEXPASTXBUTXCANNOTXCONTROLXITZXWEXCONTROLXTHEXFU
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.***