**Vanier College**

**Faculty of Science and Technology**

**Cyber Security**

**420-950-VA-1222**

**Lab 1 Report: Caesar Cipher**

**Encryption and Decryption**

**Three Team Member**

Yao Wu

Zheng Xue

Rubin Zhang

**Encryption Algorithm:** **Yao Wu**

**Decryption Algorithm:** **Zheng Xue**

**Pytest Results: Rubin Zhang**

**Teams test with no shift and Group 4 provided us encrypted message:**

Jneavat: Fngryyvgr fheirvyynapr qrgrpgf vapernfrq npgvivgl ng gur abegurea qbpx. Vavgvngr pbhagrezrnfherf ng avtugsnyy. Rapelcg nyy zvffvba ybtf jvgu xrl Fvtzn.

**Analysis:**

The provided message is using a Caesar cipher. In a Caesar cipher, each letter in the plaintext is shifted a certain number of places down or up the alphabet.

**Decryption Process:**

To decrypt the message, we will apply a brute-force approach by trying all possible shift values (0 to 25) to decipher the text. We'll analyze the context, patterns, and readability to determine the correct decryption.

Decrypted Messages (using various shift values):

Shift 0: Jneavat: Fngryyvgr fheirvyynapr qrgrpgf vapernfrq npgvivgl ng gur abegurea qbpx. Vavgvngr pbhagrezrnfherf ng avtugsnyy. Rapelcg nyy zvffvba ybtf jvgu xrl Fvtzn.

Shift 1: Imduzsu: Emfqxxufq egdhquxxmozq pqfqoef uzodqmeqp mofuhufk mf ftq zadftqdz paox. Uzufumfq oagzfqdyqmeqgdeq mf zustfrmxx. Qzodkbf mxx yueeuaz xase itq Eusym.

Shift 2: Hlctyrt: Dlepwwtep dfcgptwwlnyp opepnde tyncpldpo lnetgetj le esp yzcespdy oznw. Tytetlep nzfyepcxpldpcfzq le ytrseqlww. Pyncjae lww xtddtzy wzrd hsp DtrxL.

Shift 3: Gkbtsxqs: Ckdfovsdq cebfosvvkmxo nodomcd sxmbokcon kmdsfdsi kd dro xybdrocz nymv. Sxsdskdo myexdobwokcdbeyp kd xsqrdpkvv. Oxmbizd kvv wscssyx vyqc gro CsqwK.

Shift 4: Fjasrwpr: Bjcenhrcp bdavernujlwn mncnlbc rwlanjbmn jlcrecrh jc cqn wxacqnby mxlu. Rwrcrjcn lxdwcnavnjbcadxo jc wrpqcjjuu. Nwlahyc juu vrrbbrx uxpb fqn BrpvJ.

Shift 5: Eizrqvoq: Aibdmgqbo aczdmqtmikvm lmbmkab qvkzmialm ikbqdbgq ib bpm vwzbpmax lwkt. Qvqbqibm kwcvbmzumialzwcn ib vqopbii tt. Mvkzgxb itt uqqaaqw twoa epm AqouI.

Shift 6: Dhypupnp: Zahclfpzn zbyclpslhjul klaljza pujylhzkl hjapcafp ha aol uvyallzv kjvs. Pupaphal jvbualytlhzkyvbm ha upnoahh ss. Lujyfwa hss tppzzpv svnz dol ZpntH.

Shift 7: Cgxotomo: Yzgbkeoym yaxbkoqkgitk jkzkjyz otixkgyjk gizobzeo gz znk tuxzkkyu jiur. Otozogzk iuatzkxsikgyxual gz tomnzgg rr. Ktixevz grr sooyyou rumy cnk YomsG.

Shift 8: Bfwnsnln: Xyfajdnxl xzwajnpjfhsj ijyjixy nshwjfxij fhnaydzn fy ymj stwyjjxt ihtq. Nsnynfyj htszyjwrhjfwxtzk fy snlmyff qq. Jshwduy fqq rnnxxnt qtlx bmj XnlrF.

Shift 9: Aevmrmkm: Wxezicmwk wyvzimmoegri hixihwx mrgvigwhi egmzxcym ex xli rsxviixs hgps. Mrmxmexi grsxyivqgievwysj ex rmklxee pp. Irgvctx epp qmmwwms pskw ali WmkqE.

Shift 10: Zdulqljl: Vwdyhlvzj vxuyhllndfqh ghwhgvw lquufvgzh dflywbxl dw wkh qrwhhhwr gfpr. Lqlwlwdh fqrwxhupfhdurxri dw qljkwdd oo. Hqfubsw doo pllvvlr orjv zkh VljpD.

Shift 11: Yctpkiki: Uvcxgkuyi uwtxgkkmcep ggvgfuv kptteufyg cekxvawk cv vjg pqvgggvq feoq. Kpkvkvcg epqvwgtoegcqtwqh cv pkijvcc nn. Gpetarv cnn okkuukq nqiu yjg UkioC.

Shift 12: Xbsojhjh: Tubwfjtxh tvswfjjlbdno fufgetu josstdexf bdjwuzvj bu uif opufffup ednp. Jojujubf dpquvfsndfbpsvpg bu ojhitbb mm. Fodszqu bmm njjttjp mphthxif TjhnB.

Shift 13: Warning: Satellite surveillance detects increased activity at the northern dock. Initiate countermeasures at nightfall. Encrypt all mission logs with key Sigma.

Shift 14: Vzqmhfhf: Rszudhhrf rtqudhhjzblm dsdecrs hmqqrbcvd zbhusrth zs sgd mnrsddsnzbk. Hmhmghhz bnqsqtdqlbqdqtnedz mh mfggrzz kk. Dmxbqxc zkk lhggrrn knfmrvgc RhflZ.

Shift 15: Uyplggeg: Qrztcggqe qspcggiizakl crcdbqr glppqabuc yagtrqsg yr rfc lmqrccrmyaj. Glglfggy amprpscpkapepsmcdy lg leffqyy jj. Clwapwb yjj kgffqqm jlemqufb QgekY.

Shift 16: Txokffdf: Lqrsbffdq probffhhyzjk bqbbaqp fkkopzatx xzfsqprf xq qeb klpqbbqlxzi. Fkfkfeex zloqorbojzodorlbcx kf kdeepxx ii. Bkuzova xii jfeeppl ikdlptea PfdjX.

Shift 17: Swnjeece: Kpqraeecp oqnaeeggxyij apzaapoejjnoyzr swerpoqe wp pda jkopaaqkwyh. Ejjeeddw yknpnqankyncnqkabw je jcddoww hh. Ajtynuz whh ieddook hjckosdz OeciW.

Shift 18: Rvmiddbd: Jopqdddbo npmzddffxhi izyzzondiimmxyq rvdqonpd vo ocz ijnozzpjvxg. Didddccv xjmompzmjxmmbpjzav id ibccnvv gg. Zisxnty vgg hdccnnj gibjnrcy Nd bh S v

Shift 19: Qulhccac: Inopcccanoly hxyyynmchhllkxwp qucpnmoc un nby himnnoiuwuf. Chcccbbu wilnlolyiwillzoyzu hc haabmuu ff. Yhrwmsx uff gcbblmi fhahmqbx McagU.

Shift 20: Tjkgbazz: Hmnoabbznmkxgwwov tbolmnbz tm maw ghllmmhvtve. Bgbbbbat vhkmknkhvvhkkynxyt gb ggzallt ee. Xgqvlrw tee fbaaklh egzglpay LbzfT.

Shift 21: Silfazzy: Glmnzaaymljwfvvnu sanklmay sl lzv fgkkllgusud. Afazzazs ugjljmjguugjjxmwxs fa ffyzkks dd. Wfpukqv sdd eazzjkg dfyfkozx KayeS.

Shift 22: Rhkeyyxx: Flkmyzzxlkiveuutm rzmjklzx rk kyu efjjkkftrtc. Zezyyzyr tfikliffttfillwlvr ez eexyjjr cc. Veotjpu rcc dzyyijf cexejnyw JzxdR.

Shift 23: Qgjdxxww: Ekjlxyywkjhtdttsl qylijkyw qj jxt deiijjesqsb. Ydyxxxyq sghjkheessgghkkukuq dy ddwxiiq bb. Udnsiot qbb cyxxhie bdwdimxv IwzcQ.

Shift 24: Pfiwvwwv: Djikwxxvjigcscssk pxkhijxv pi iws cdhhiddrpza. Xcxxwwxp rfgijgddrrffgjjtjtp cx ccvwhhp aa. Tcmrhns paa bxxwhhd acvchlwu HvybP.

Shift 25: Oehvuvvu: Cihjvwwuihfbrrbrj owjghiwu oh hvr bcggghcqoyz. Wbwwvvwo qefhifcqqeefiiissio bw bbugggo zz. Sblqgmr ozz awwvggc zbubgkvt GuxaO.

Enter a message to encrypt: Jneavat: Fngryyvgr fheirvyynapr qrgrpgf vapernfrq npgvivgl ng gur abegurea qbpx. Vavgvngr pbhagrezrnfherf ng avtugsnyy. Rapelcg nyy zvffvba ybtf jvgu xrl Fvtzn.

Enter shift value: 13

Original Message: Jneavat: Fngryyvgr fheirvyynapr qrgrpgf vapernfrq npgvivgl ng gur abegurea qbpx. Vavgvngr pbhagrezrnfherf ng avtugsnyy. Rapelcg nyy zvffvba ybtf jvgu xrl Fvtzn.

Encrypted Message: Warning: Satellite surveillance detects increased activity at the northern dock. Initiate countermeasures at nightfall. Encrypt all mission logs with key Sigma.

Decrypted Message: Jneavat: Fngryyvgr fheirvyynapr qrgrpgf vapernfrq npgvivgl ng gur abegurea qbpx. Vavgvngr pbhagrezrnfherf ng avtugsnyy. Rapelcg nyy zvffvba ybtf jvgu xrl Fvtzn.

**Conclusion:**

By attempting various shift values and analyzing the results, it appears that the original message was encrypted with a Caesar cipher using a shift of 13. The decrypted message is: " Warning: Satellite surveillance detects increased activity at the northern dock. Initiate countermeasures at nightfall. Encrypt all mission logs with key Sigma."

This is the encryption and decryption main program.

def encrypt(message, shift):  
 encrypted\_message = ""  
 for char in message:  
 if char.isalpha():  
 if char.isupper():  
 encrypted\_message += chr((ord(char) + shift - 65) % 26 + 65)  
 else:  
 encrypted\_message += chr((ord(char) + shift - 97) % 26 + 97)  
 else:  
 encrypted\_message += char  
 return encrypted\_message  
  
def decrypt(message, shift):  
 decrypted\_message = ""  
 for char in message:  
 if char.isalpha():  
 if char.isupper():  
 decrypted\_message += chr((ord(char) - shift - 65) % 26 + 65)  
 else:  
 decrypted\_message += chr((ord(char) - shift - 97) % 26 + 97)  
 else:  
 decrypted\_message += char  
 return decrypted\_message  
  
def main():  
 # Get input from the user  
 message = input("Enter a message to encrypt: ")  
 shift = int(input("Enter shift value: "))  
  
 # Encrypt and decrypt the message  
 encrypted\_message = encrypt(message, shift)  
 decrypted\_message = decrypt(encrypted\_message, shift)  
  
 # Print the results  
 print("\nOriginal Message:", message)  
 print("Encrypted Message:", encrypted\_message)  
 print("Decrypted Message:", decrypted\_message)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

This is the encryption and decryption test program.

import unittest

from main import encrypt, decrypt

class TestCryptoFunctions(unittest.TestCase):

def test\_encrypt(self):

# Test encryption with various cases

self.assertEqual(encrypt("HELLO", 3), "KHOOR")

self.assertEqual(encrypt("hello", 3), "khoor")

self.assertEqual(encrypt("Hello World", 3), "Khoor Zruog")

self.assertEqual(encrypt("Hello, World!", 3), "Khoor, Zruog!")

self.assertEqual(encrypt("HELLO", -3), "EBIIL")

self.assertEqual(encrypt("HELLO", 30), "LIPPS")

self.assertEqual(encrypt("", 3), "")

def test\_decrypt(self):

# Test decryption with various cases

self.assertEqual(decrypt("KHOOR", 3), "HELLO")

self.assertEqual(decrypt("khoor", 3), "hello")

self.assertEqual(decrypt("Khoor Zruog", 3), "Hello World")

self.assertEqual(decrypt("Khoor, Zruog!", 3), "Hello, World!")

self.assertEqual(decrypt("EBIIL", -3), "HELLO")

self.assertEqual(decrypt("LIPPS", 30), "HELLO")

self.assertEqual(decrypt("", 3), "")

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

This is the brute force decryption program.

def caesar\_cipher(text, shift, mode="encrypt"):

result = ""

for char in text:

if char.isalpha():

is\_upper = char.isupper()

base = ord("A") if is\_upper else ord("a")

if mode == "encrypt":

result += chr((ord(char) - base + shift) % 26 + base)

elif mode == "decrypt":

result += chr((ord(char) - base - shift) % 26 + base)

else:

result += char

return result

def brute\_force\_decrypt(message):

for shift in range(26):

decrypted\_message = caesar\_cipher(message, shift, mode="decrypt")

print(f"Shift {shift}: {decrypted\_message}")

def main():

# Get input from the user

message = input("Enter a message: ")

# Encrypt the message

shift\_value = 0

encrypted\_message = caesar\_cipher(message, shift\_value, mode="encrypt")

# Brute-force decrypt and print the results

print("\nOriginal Message:", message)

print("Encrypted Message:", encrypted\_message)

print("\nBrute-force Decryption:")

brute\_force\_decrypt(encrypted\_message)

if \_\_name\_\_ == "\_\_main\_\_":

main()