	Hope Foundation's, **Finolex Academy of Management and Technology, Ratnagiri**		
	Department of Information Technology		
Subject name: SECURITY LAB			Subject Code: ITL502
Class	TE IT	Semester – V REV 2019 C	Academic year: 2021-22
Name of Student	**Garde Tanmay Pramod**	**QUIZ Score :**	
Roll No	**12**	Assignment/Experiment No.	02
Title**:** **Design and Implementation of product cipher using Substitution and Transposition ciphers**			

1. **Course objectives applicable:**
   LOB1- To be able to apply the knowledge of symmetric cryptography to implement simple Ciphers.

2. **Course outcomes applicable:**
   LO1- Apply the knowledge of symmetric cryptography to implement simple ciphers.

3. Learning Objectives:

- To conceal the context of some message from all , except the sender and recipient (privacy or secrecy) to prevent eavesdropping.
- To verify the correctness of a message to the recipient (authentication) to prevent tampering.

3. **Practical applications of the assignment/experiment:**

- It helps to provide accountability, fairness, accuracy and confidentiality.
- It can prevent fraud in electronic commerce and assure the validity of financial transactions.
- It can prove one's identity and protect one's anonymity.

5. Prerequisites: Understanding working of cryptosystem.

6. Hardware Requirements:

1. PC with 4GB RAM, 500GB HDD,

7. Software Requirements:

1. Programming language C, C++, Java

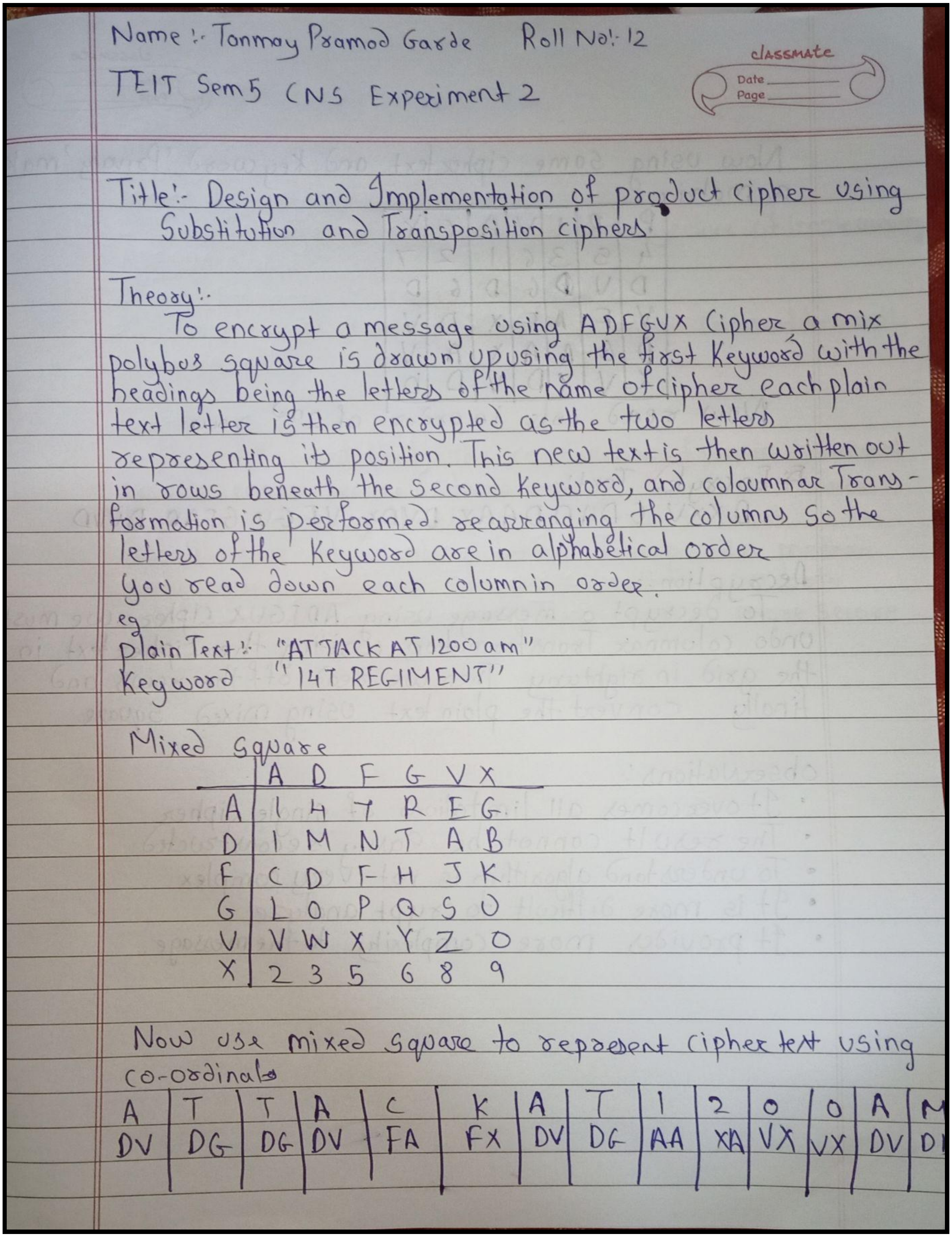
8. Quiz Questions (if any): (Online Exam will be taken separately batchwise, attach the certificate/ Marks obtained)

1. What is Symmetric Key cryptography?
2. What is Asymmetric Key cryptography?
3. Compare Substitution and Transposition ciphers.

9. Experiment/Assignment Evaluation:

Sr. No.	**Parameters**		**Marks obtained**	**Out of**
1	Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used -			6
2	Neatness/presentation			2
3	Punctuality			2
Date of performance (DOP)	**20/7/2021**	**Total marks obtained**		**10**
Date of checking (DOC)		**Signature of teacher**		

Theory: <HANDWRITTEN>



Name :- Tonmay Pramod Garde Roll No:- 12

TEIT Sem 5 CNS Experiment 2

classmate
Date
Page

Title:- Design and Implementation of product cipher using Substitution and Transposition ciphers

Theory:-

To encrypt a message using ADFGVX Cipher a mix polybus square is drawn upusing the first Keyword with the headings being the letters of the name ofcipher each plain text letter is then encrypted as the two letters representing its position. This new textis then written out in rows beneath the second keyword, and coloumnar Transformation is performed rearranging the columns so the letters of the Keyword are in alphabetical order you read down each columnin order.

eg

plain Text :- "ATTACK AT 1200 am"

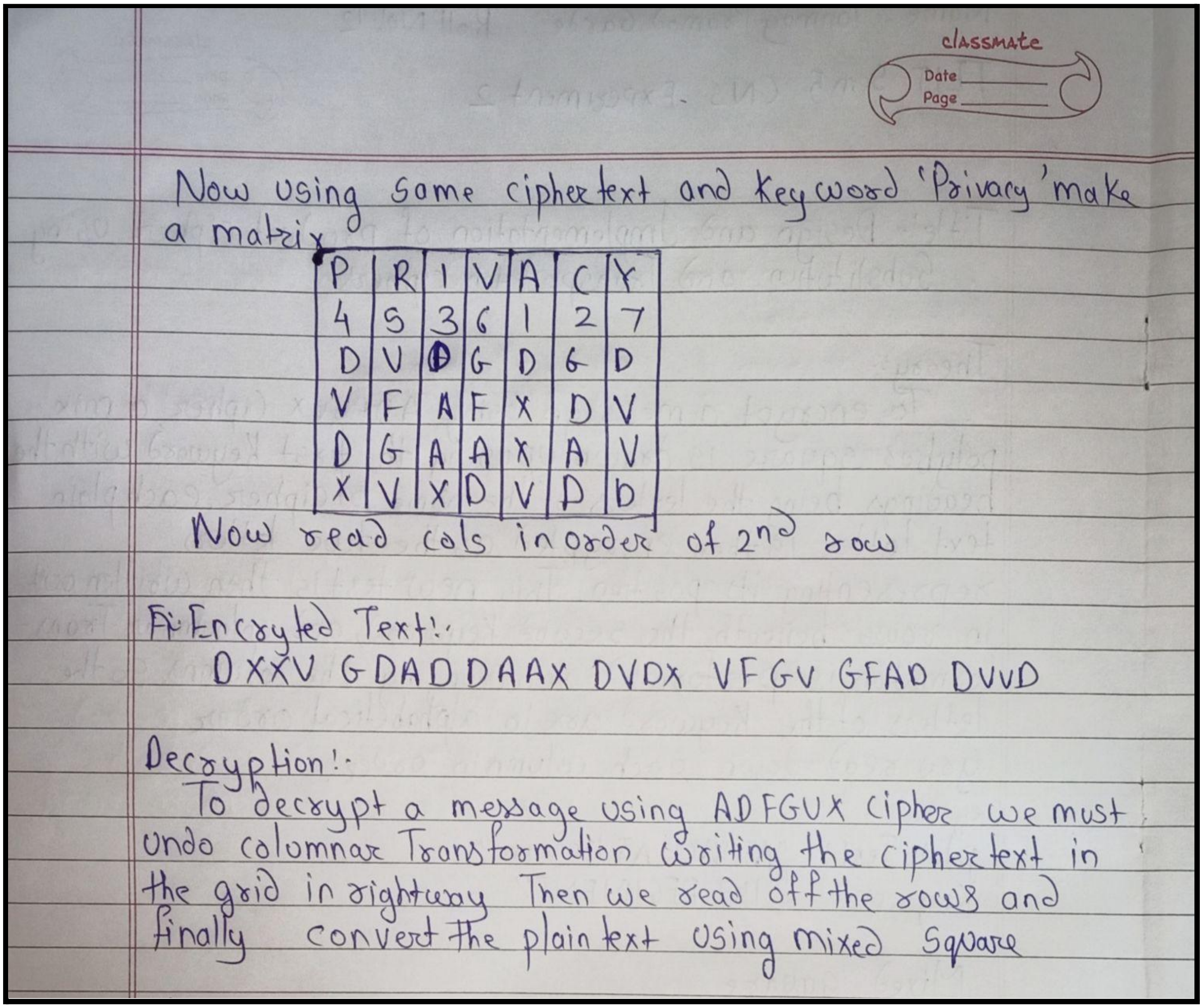
Keyword "14T REGIMENT"

Mixed Square

	A	D	F	G	V	X
A	1	4	7	R	E	G
D	I	M	N	T	A	B
F	C	D	F	H	J	K
G	L	O	P	Q	S	U
V	V	W	X	Y	Z	O
X	2	3	5	6	8	9

Now use mixed square to represent cipher text using co-ordinates

A	T	T	A	C	K	A	T	1	2	0	0	A	M
DV	DG	DG	DV	FA	FX	DV	DG	AA	XA	VX	VX	DV	D



classmate

Date

Page

Now using same ciphertext and Keyword 'Privacy' make a matrix

P	R	I	V	A	C	Y
4	5	3	6	1	2	7
D	V	D	G	D	G	D
V	F	A	F	X	D	V
D	G	A	A	X	A	V
X	V	X	D	V	D	D

Now read cols in order of 2nd row

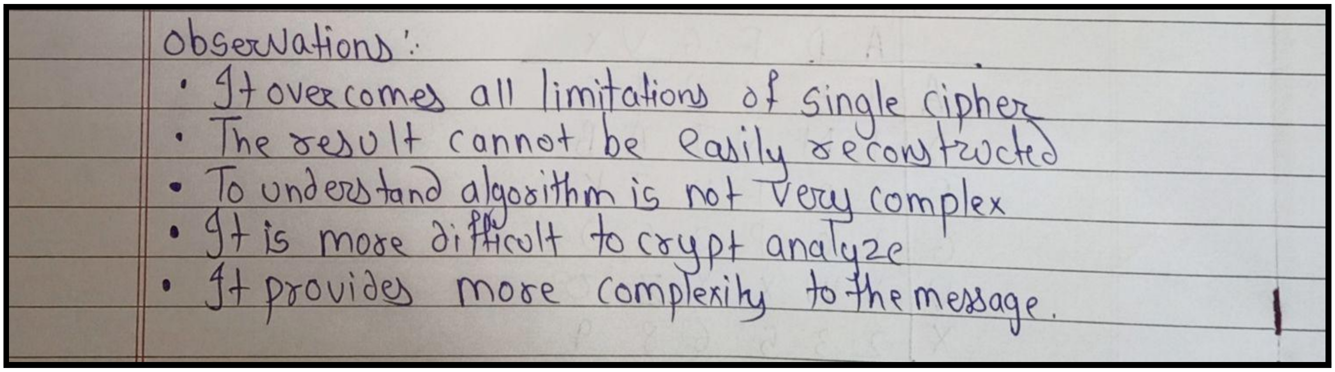
~~F~~ Encryted Text:

DXXV GDAD DAAX DVDX VFGV GFAD DVVD

Decryption:

To decrypt a message using ADFGVX cipher we must undo columnar Transformation writing the ciphertext in the grid in rightway Then we read off the rows and finally convert the plain text using mixed Square

Observations: <HANDWRITTEN>



Observations:

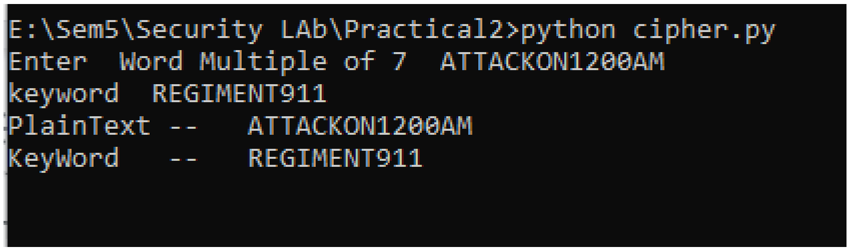
- It overcomes all limitations of single cipher
- The result cannot be easily reconstructed
- To understand algorithm is not very complex
- It is more difficult to crypt analyze
- It provides more complexity to the message.

Program:

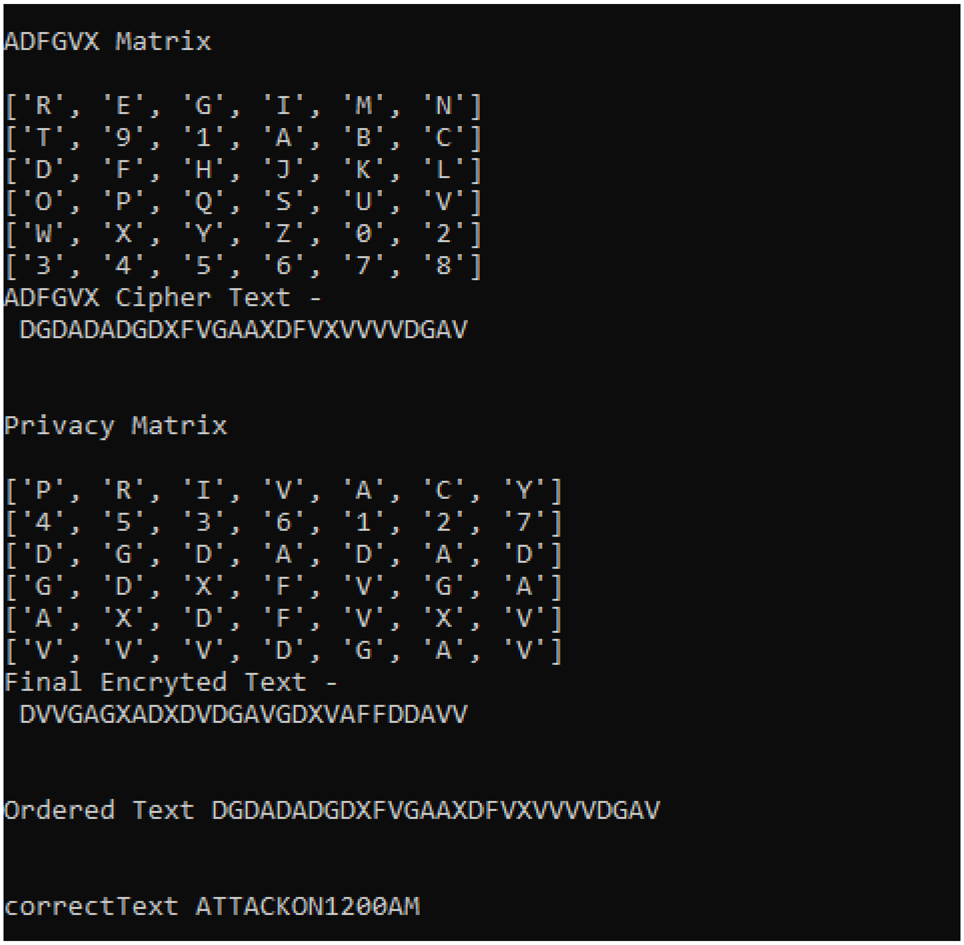
```
plainText = input("Enter  Word Multiple of 7  ")
# Any 14 char Word eg ATTACKON1200AM
keywords= input("keyword  ")
Keywords=keywords
# Make ADFGVX Matrix
seentext = []
ADVGVX_MATRIX = []

def makeADFGVXMatrix ():
  # Assuming Keyword doesnt have repetitions
    global ADVGVX_MATRIX,seentext,keywords
    for i in range(0,6):
      newArr=[]
      startAscii = 65
      for j in range(0,6):
        while (keywords!="" and keywords[0] in seentext) :
            keywords=keywords[1:]

        if keywords != "" :
          newArr.append(keywords[0])
          seentext.append(keywords[0])
          keywords=keywords[1:]

        else :
          if startAscii > 90 :
            startAscii=48

          while (chr(startAscii) in seentext) :
            startAscii+=1
            if startAscii > 90 :
              startAscii=48
          newArr.append(chr(startAscii))
          seentext.append(chr(startAscii))
          startAscii+=1
      ADVGVX_MATRIX.append(newArr)
```

```
makeADFGVXMatrix()

positions=["A","D","F","G","V","X"]
cipher_text=""
def search (mat,char):
  global cipher_text,positions
  for i in range(0,6):
    for j in range(0,6):
      if char==mat[i][j]:

        cipher_text+=positions[i]
        cipher_text+=positions[j]

def getcipherText (plainText,mat):
  global cipher_text,positions
  for i in plainText :
    search(mat,i)

getcipherText(plainText,ADVGVX_MATRIX)
privacy=["P","R","I","V","A","C","Y"]

PrivacyMatrix = [["P","R","I","V","A","C","Y"]]

key_sort =privacy
indices=[]
key_sort.sort()
for i in PrivacyMatrix[0]:
  indices.append(str(key_sort.index(i)+1))
PrivacyMatrix.append(indices)
# print(PrivacyMatrix)

def makePrivacyMatrix(cipher_text):
  global PrivacyMatrix
  temp=[]
  for i in range(0,len(cipher_text)):
    temp.append(cipher_text[i])
    if len(temp)==7:
      PrivacyMatrix.append(temp)
      temp=[]
makePrivacyMatrix(cipher_text)
currentCol=1
def selectCol (mat) :
  global currentCol
  for i in range(0,len(mat)) :
    if currentCol==int(mat[i]) :
      currentCol+=1
      return int(i)


FinalText = ""

def readPrivacyMatrix(mat):
  global FinalText
  n=len(mat)
  m=len(mat[0])
  # print(n,m)
  for i in range(0,m):
    whichCol=selectCol(mat[1])
    for j in range(2,n):
      FinalText=FinalText+mat[j][whichCol]

def printMatrix(mat) :
  for i in mat :
    print(i)
    # for j in mat[i]:
    #   print(i,end=" ")
    # print("")

readPrivacyMatrix(PrivacyMatrix)
print("PlainText --  ",plainText)
print("KeyWord   --  ",Keywords)
print("\n\n\n\nADFGVX Matrix \n")
printMatrix(ADVGVX_MATRIX)

print("ADFGVX Cipher Text -\n",cipher_text)

print("\n\nPrivacy Matrix \n")
printMatrix(PrivacyMatrix)
print("Final Encryted Text - \n",FinalText)




def decryption (ADFGVX_MATRIX,text,keyword):
  indexing = ['A','D','F','G','V','X']
  # P R I V A C Y
  keyword_list =[]
  for i in keyword :
    keyword_list.append(i)
  newKeyword=keyword_list.copy()
  newKeyword.sort()
  ordered_text = ""
  temp_mat = []
  for i in keyword_list :
    position = newKeyword.index(i)
    length=(len(text)//len(keyword_list))
    tempStr = text[position*length:position*length+length]
    temp_mat.append(tempStr)
  for i in range(0,len(temp_mat[0])) :
    for j in range(0,len(temp_mat)):
      ordered_text=ordered_text+temp_mat[j][i]
  print("\n\nOrdered Text",ordered_text)
  correctText=""
  for i in range (0,len(ordered_text)//2) :
    row_char=ordered_text[i*2]
    col_char=ordered_text[i*2+1]
    row = indexing.index(row_char)

    col = indexing.index(col_char)


    correctText = correctText + ADFGVX_MATRIX[row][col]

  print("\n\ncorrectText",correctText)

decryption(ADVGVX_MATRIX,FinalText,"PRIVACY")
```

Results:

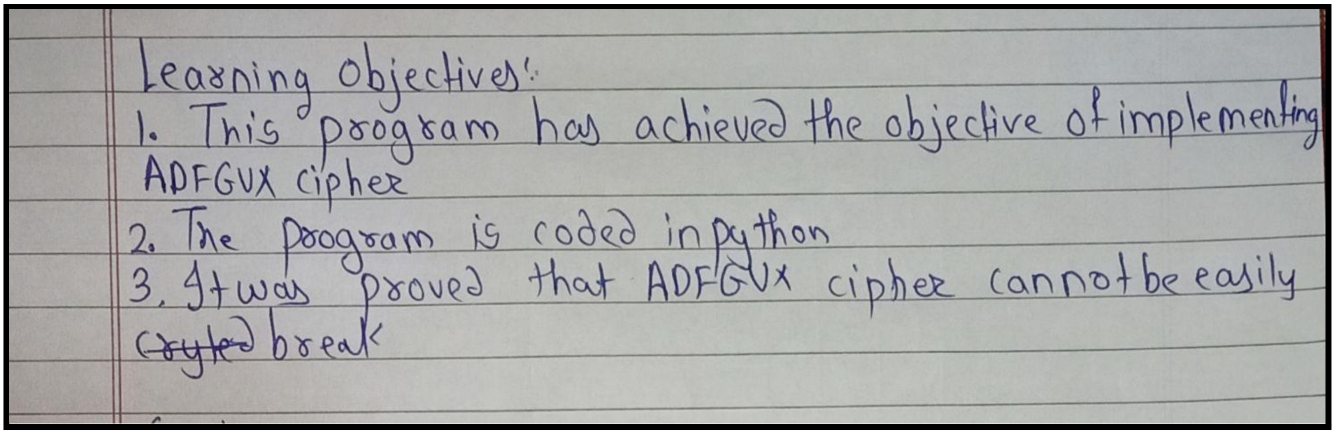


```
E:\Sem5\Security LAb\Practical2>python cipher.py
Enter  Word Multiple of 7  ATTACKON1200AM
keyword  REGIMENT911
PlainText --   ATTACKON1200AM
KeyWord   --   REGIMENT911
```



```
ADFGVX Matrix

['R', 'E', 'G', 'I', 'M', 'N']
['T', '9', '1', 'A', 'B', 'C']
['D', 'F', 'H', 'J', 'K', 'L']
['O', 'P', 'Q', 'S', 'U', 'V']
['W', 'X', 'Y', 'Z', '0', '2']
['3', '4', '5', '6', '7', '8']
ADFGVX Cipher Text -
 DGDADADGDXFVGAAXDFVXVVVVDGAV


Privacy Matrix

['P', 'R', 'I', 'V', 'A', 'C', 'Y']
['4', '5', '3', '6', '1', '2', '7']
['D', 'G', 'D', 'A', 'D', 'A', 'D']
['G', 'D', 'X', 'F', 'V', 'G', 'A']
['A', 'X', 'D', 'F', 'V', 'X', 'V']
['V', 'V', 'V', 'D', 'G', 'A', 'V']
Final Encryted Text -
 DVVGAGXADXDVDGAVGDXVAFFDDAVV


Ordered Text DGDADADGDXFVGAAXDFVXVVVVDGAV


correctText ATTACKON1200AM
```

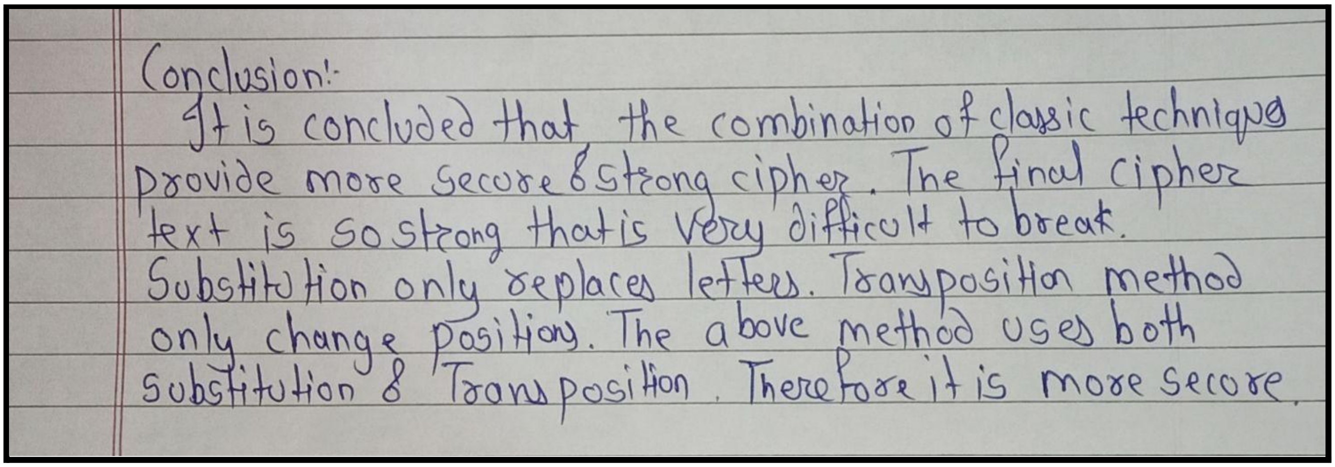
Learning Outcomes Achieved <HANDWRITTEN>



Learning objectives:-

1. This program has achieved the objective of implementing ADFGVX cipher
2. The program is coded in python
3. It was proved that ADFGVX cipher cannot be easily ~~Cryted~~ break

Conclusion: <HANDWRITTEN>



Conclusion:-

It is concluded that the combination of classic techniques provide more secure & strong cipher. The final cipher text is so strong that is very difficult to break. Substitution only replaces letters. Transposition method only change positions. The above method uses both substitution & Transposition. Therefore it is more secure.

References :

1. Build your own Security Lab, Michael Gregg, Wiley India.

2. CCNA Security, Study Guide, TIm Boyles, Sybex.