

Cryptography and Network Security

Lab Assignment - I

Program 1: Caesar Cipher

Pseudocode/Algorithm

- **Start:** Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice:** Take the user's choice for encryption, decryption, or exiting the program.
- **Exit Check:** If the choice is to exit, terminate the program.
- **Input Text and Key:** Prompt the user to enter the plaintext and the key.
- **Encrypt/Decrypt:** Based on the choice, call the caesar function with the appropriate mode ('e' for encryption, 'd' for decryption).
- **Display Result:** Print the result of the encryption or decryption.
- **Repeat or End:** Return to step 1 unless the user chooses to exit.

Code

```
def caesar(text, key, chk):  
    result=""  
    for i in text:  
        if i==' ':  
            result+=' '  
            continue  
        if chk == 'e':  
            result+=chr((ord(i)+key-97)%26+97)  
        elif chk=='d':  
            result+=chr((ord(i)-key-97)%26+97)  
    return result  
  
print("Siddhanth Monnappa\t22BCE3061")  
while(True):  
    print("--CAESAR CIPHER--")
```

```
print("1) Encrypt Text")
print("2) Decrypt Text")
print("3) Exit")
ch=input("Enter your choice: ")
if ch=='3':
    break
text=input("Enter PlainText: ")
text=text.lower()
key=int(input("Enter Key: "))
if ch=='1':
    print(f"Caesar Cipher: {caesar(text, key, 'e')}")
elif ch=='2':
    print(f"Decyphered PlainText: {caesar(text, key, 'd')}")
```

Code Screenshot

```

1  def caesar(text, key, chk):
2      result=""
3      for i in text:
4          if i==' ':
5              result+=' '
6              continue
7          if chk == 'e':
8              result+=chr((ord(i)+key-97)%26+97)
9          elif chk=='d':
10             result+=chr((ord(i)-key-97)%26+97)
11     return result
12
13     print("Siddhanth Monnappa\t22BCE3061")
14     while(True):
15         print("--CAESAR CIPHER--")
16         print("1) Encrypt Text")
17         print("2) Decrypt Text")
18         print("3) Exit")
19         ch=input("Enter your choice: ")
20         if ch=='3':
21             break
22         text=input("Enter PlainText: ")
23         text=text.lower()
24         key=int(input("Enter Key: "))
25         if ch=='1':
26             print(f"Caesar Cipher: {caesar(text, key, 'e')}")
27         elif ch=='2':
28             print(f"Decyphered PlainText: {caesar(text, key, 'd')}")

```

Output

```

Siddhanth Monnappa      22BCE3061
--CAESAR CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter your choice: 1
Enter PlainText: Sentence
Enter Key: 5
Caesar Cipher: xjsyjshj
--CAESAR CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter your choice: 2
Enter PlainText: xjsyjshj
Enter Key: 5
Decyphered PlainText: sentence
--CAESAR CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter your choice: 3

```

Program 2: Playfair Cipher

Pseudocode/Algorithm

- **Start:** Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice:** Take the user's choice for encryption, decryption, or exit.
- **Exit Check:** If the choice is to exit, terminate the program.
- **Generate Key Matrix:** Take the user-input key, preprocess it, and generate a 5x5 key matrix by combining the key and the alphabet, excluding duplicates and replacing 'j' with 'i'.
- **Preprocess Text:** Preprocess the input text by converting it to lowercase, removing spaces, replacing 'j' with 'i', and appending 'x' if the text length is odd. Split the text into digraphs.
- **Encrypt/Decrypt:** Based on the choice, process each digraph using the encrypt or decrypt function, applying Playfair rules for rows, columns, and rectangles.
- **Display Result:** Print the resulting ciphertext (for encryption) or plaintext (for decryption) and return to step 1 unless the user chooses to exit.

Code

```
def encrypt(key_matrix, d):
```

```
    a, b = d
```

```
    row_a, col_a = divmod(key_matrix.index(a), 5)
```

```
    row_b, col_b = divmod(key_matrix.index(b), 5)
```

```
    if row_a == row_b:
```

```
        col_a = (col_a + 1) % 5
```

```
        col_b = (col_b + 1) % 5
```

```
    elif col_a == col_b:
```

```
        row_a = (row_a + 1) % 5
```

```
        row_b = (row_b + 1) % 5
```

```
    else:
```

```
        col_a, col_b = col_b, col_a
```

```
return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]
```

```
def decrypt(key_matrix, d):
```

```
    a, b = d
```

```
    row_a, col_a=divmod(key_matrix.index(a), 5)
```

```
    row_b, col_b=divmod(key_matrix.index(b), 5)
```

```
    if row_a==row_b:
```

```
        col_a=(col_a-1)%5
```

```
        col_b=(col_b-1)%5
```

```
    elif col_a==col_b:
```

```
        row_a=(row_a-1)%5
```

```
        row_b=(row_b-1)%5
```

```
    else:
```

```
        col_a, col_b=col_b, col_a
```

```
    return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]
```

```
print("Siddhanth Monnappa\t22BCE3061\n")
```

```
while(True):
```

```
    print("--PLAYFAIR ENCRYPTION--")
```

```
    print("1) Encrypt Text")
```

```
    print("2) Decrypt Text")
```

```
    print("3) Exit")
```

```
    ch=int(input("Enter Choice: "))
```

```
if ch==3:
    break
text=input("Enter Text: ")
key=input("Enter Key: ")

alpha='abcdefghijklmnopqrstuvwxyz'
key=key.lower().replace(' ', '').replace('j', 'i')

key_matrix=""
for alp in key+alpha:
    if alp not in key_matrix:
        key_matrix+=alp

text=text.lower().replace(' ', '').replace('j', 'i')
if len(text)%2==1:
    text+='x'

digraph=[text[i:i+2] for i in range(0, len(text), 2)]

result=""
if(ch==1):
    for d in digraph:
        result+=encrypt(key_matrix,d)
    print("Ciphertext:", result)
elif(ch==2):
    for d in digraph:
```

```

result+=decrypt(key_matrix,d)

print("Decrypted text:", result)

```

Code Screenshot

```

1 def encrypt(key_matrix, d):
2     a, b = d
3     row_a, col_a=divmod(key_matrix.index(a), 5)
4     row_b, col_b=divmod(key_matrix.index(b), 5)
5
6     if row_a==row_b:
7         col_a=(col_a+1)%5
8         col_b=(col_b+1)%5
9     elif col_a==col_b:
10        row_a=(row_a+1)%5
11        row_b=(row_b+1)%5
12    else:
13        col_a, col_b=col_b, col_a
14
15    return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]
16
17 def decrypt(key_matrix, d):
18     a, b = d
19     row_a, col_a=divmod(key_matrix.index(a), 5)
20     row_b, col_b=divmod(key_matrix.index(b), 5)
21
22     if row_a==row_b:
23         col_a=(col_a-1)%5
24         col_b=(col_b-1)%5
25     elif col_a==col_b:
26         row_a=(row_a-1)%5
27         row_b=(row_b-1)%5
28    else:
29        col_a, col_b=col_b, col_a
30
31    return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]
32
33 print("Siddhanth Monnappa\t22BCE3061\n")
34 while(True):
35     print("--PLAYFAIR ENCRYPTION--")
36     print("1) Encrypt Text")
37     print("2) Decrypt Text")
38     print("3) Exit")
39     ch=int(input("Enter Choice: "))
40     if ch==3:
41         break
42     text=input("Enter Text: ")
43     key=input("Enter Key: ")
44
45     alpha='abcdefghijklmnopqrstuvwxyz'
46     key=key.lower().replace(' ', '').replace('j', 'i')
47
48     key_matrix=''
49     for alp in key+alpha:
50         if alp not in key_matrix:
51             key_matrix+=alp
52
53     text=text.lower().replace(' ', '').replace('j', 'i')
54     if len(text)%2==1:
55         text+='x'
56
57     digraph=[text[i:i+2] for i in range(0, len(text), 2)]
58
59     result=''
60     if(ch==1):
61         for d in digraph:
62             result+=encrypt(key_matrix,d)
63         print("Ciphertext:", result)
64     elif(ch==2):
65         for d in digraph:
66             result+=decrypt(key_matrix,d)
67         print("Decrypted text:", result)
68

```

Output

```

Siddhanth Monnappa      22BCE3061

--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter Text: EncryptThis
Enter Key: boss
Ciphertext: hksuvtuudnfs
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter Text: hksuvtuudnfs
Enter Key: boss
Decrypted text: encryptthisx
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter Text: sentence
Enter Key: passed
Ciphertext: edmudmga
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter Text: edmudmga
Enter Key: passed
Decrypted text: sentence
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3

```

Program 3: Hill Cipher

Pseudocode/Algorithm

- **Start:** Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice:** Take the user's choice for encryption, decryption, or exit.
- **Exit Check:** If the choice is to exit, terminate the program.
- **Input Key:** Prompt the user for an alphabetic key of at least 4 characters.
- **Input Text:** Prompt the user to encrypt or decrypt.
- **Format Text:** Preprocess the text by converting it to uppercase, removing spaces, and appending "X" if the length is odd.
- **Key Matrix:** Generate a 2x2 key matrix from the first 4 characters of the key, converting letters to numeric values.
- **Encrypt/Decrypt Matrix:** If decrypting, calculate the inverse of the key matrix. If the inverse doesn't exist, terminate with an error.
- **Text to Numbers:** Convert the text into numeric values using their alphabetic positions.
- **Matrix Multiplication:** Multiply the numeric text with the key matrix (for encryption) or its inverse (for decryption), modulo 26.
- **Numbers to Text:** Convert the resulting numbers back to text using their alphabetic positions.
- **Display Result:** Print the resulting ciphertext (for encryption) or plaintext (for decryption) and return to step 1 unless the user chooses to exit.

Code

```
def format_text(text):
    text=text.upper().replace(" ", "")
    if len(text)%2!=0:
        text+="X"
    return text
```

```
def numeric_value(text):
    numbers=[]
    for char in text:
        if char.isalpha():
```



```
        numbers.append(ord(char)-ord("A"))  
    return numbers
```

```
def key_to_matrix(key):  
    key=key.upper().replace(" ", "")  
    if len(key) < 4:  
        print("Key Must be atleast 4 characters long")  
        exit()  
    key=key[:4]  
    key_numbers=numeric_value(key)  
    return [[key_numbers[0], key_numbers[1]],[key_numbers[2], key_numbers[3]],]
```

```
def num_to_text(num):  
    text=""  
    for n in num:  
        text+=chr(n+ord("A"))  
    return text
```

```
def mod_inverse(a, m):  
    for i in range(1, m):  
        if (a*i)%m==1:  
            return i  
    return None
```

```
def matrix_multiply(matrix1, matrix2):  
    result=[]
```

```

for i in range(0, len(matrix1), 2):
    row_result=[]
    for j in range(2):
        result_val=sum(matrix1[i+x]*matrix2[x][j] for x in range(2))%26
        row_result.append(result_val)
    result.extend(row_result)
return result

```

```

def inverse_matrix(key_matrix):
    det=(key_matrix[0][0]*key_matrix[1][1]-key_matrix[0][1]*key_matrix[1][0])%26
    inv_det=mod_inverse(det, 26)
    if inv_det is None:
        return None

    adj_mat=[[key_matrix[1][1], -key_matrix[0][1]],[-key_matrix[1][0], key_matrix[0][0]],]

    for i in range(2):
        for j in range(2):
            adj_mat[i][j]=adj_mat[i][j]%26

    inv_mat=[[((inv_det*adj_mat[i][j]))%26 for j in range(2)]for i in range(2)]
    return inv_mat

```

```

def encrypt(text, key):
    text=format_text(text)
    key_matrix=key_to_matrix(key)

```

```
num=numeric_value(text)
result_numbers=matrix_multiply(num, key_matrix)
result_text=num_to_text(result_numbers)
return result_text
```

```
def decrypt(text, key):
    text=format_text(text)
    key_matrix=key_to_matrix(key)
    key_matrix=inverse_matrix(key_matrix)
    if key_matrix is None:
        return "Key matrix is not invertible"
    num=numeric_value(text)
    result_numbers=matrix_multiply(num, key_matrix)
    result_text=num_to_text(result_numbers)
    return result_text
```

```
print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
    print("--HILL CIPHER--")
    print("1) Encrypt Text")
    print("2) Decrypt Text")
    print("3) Exit")
    ch=int(input("Enter Choice: "))
    if ch==3:
        break
    key = input("Enter an alphabetic key (min 4 characters): ")
```

```

text = input("Enter the text: ")

if (ch==1):

    print(f"Ciphertext: {encrypt(text, key)}")

elif (ch==2):

    print(f"Decrypted Text: {decrypt(text, key)}")

```

Code Screenshot

```

1 def format_text(text):
2     text=text.upper().replace(" ", "")
3     if len(text)%2!=0:
4         text+="X"
5     return text
6
7 def numeric_value(text):
8     numbers=[]
9     for char in text:
10        if char.isalpha():
11            numbers.append(ord(char)-ord("A"))
12    return numbers
13
14 def key_to_matrix(key):
15     key=key.upper().replace(" ", "")
16     if len(key) < 4:
17         print("Key Must be atleast 4 characters long")
18         exit()
19     key=key[:4]
20     key_numbers=numeric_value(key)
21     return [[key_numbers[0], key_numbers[1]], [key_numbers[2], key_numbers[3]]]
22
23 def num_to_text(num):
24     text=""
25     for n in num:
26         text+=chr(n+ord("A"))
27     return text
28
29 def mod_inverse(a, m):
30     for i in range(1, m):
31         if (a*i)%m==1:
32             return i
33     return None
34
35 def matrix_multiply(matrix1, matrix2):
36     result=[]
37     for i in range(0, len(matrix1), 2):
38         row_result=[]
39         for j in range(2):
40             result_val=sum(matrix1[i+x]*matrix2[x][j] for x in range(2))%26
41             row_result.append(result_val)
42         result.extend(row_result)
43     return result
44
45 def inverse_matrix(key_matrix):
46     det=(key_matrix[0][0]*key_matrix[1][1]-key_matrix[0][1]*key_matrix[1][0])%26
47     inv_det=mod_inverse(det, 26)
48     if inv_det is None:
49         return None
50
51     adj_mat=[[key_matrix[1][1], -key_matrix[0][1]], [-key_matrix[1][0], key_matrix[0][0]]]
52
53     for i in range(2):
54         for j in range(2):
55             adj_mat[i][j]=adj_mat[i][j]*26
56
57     inv_mat=[[(inv_det*adj_mat[i][j])%26 for j in range(2)] for i in range(2)]
58     return inv_mat
59
60 def encrypt(text, key):
61     text=format_text(text)
62     key_matrix=key_to_matrix(key)
63     num=numeric_value(text)
64     result_numbers=matrix_multiply(num, key_matrix)
65     result_text=num_to_text(result_numbers)
66     return result_text
67
68 def decrypt(text, key):
69     text=format_text(text)
70     key_matrix=key_to_matrix(key)
71     key_matrix=inverse_matrix(key_matrix)
72     if key_matrix is None:
73         return "Key matrix is not invertible"
74     num=numeric_value(text)
75     result_numbers=matrix_multiply(num, key_matrix)
76     result_text=num_to_text(result_numbers)
77     return result_text
78
79 print("Siddhanth Monnappa\t22BCE3061\n")
80 while(True):
81     print("--HILL CIPHER--")
82     print("1) Encrypt Text")
83     print("2) Decrypt Text")
84     print("3) Exit")
85     ch=int(input("Enter Choice: "))
86     if ch==3:
87         break
88     key = input("Enter an alphabetic key (min 4 characters): ")
89     text = input("Enter the text: ")
90     if (ch==1):
91         print(f"Ciphertext: {encrypt(text, key)}")
92     elif (ch==2):
93         print(f"Decrypted Text: {decrypt(text, key)}")
94

```

Output

```

Siddhanth Monnappa      22BCE3061

--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter an alphabetic key (min 4 characters): test
Enter the text: Encrypt
Ciphertext: YDGYRVY
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter an alphabetic key (min 4 characters): test
Enter the text: YDGYRVY
Decrypted Text: ENCRYPTX
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3

```

```

Siddhanth Monnappa      22BCE3061

--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter an alphabetic key (min 4 characters): road
Enter the text: sentence
Ciphertext: UENFQRIQ
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter an alphabetic key (min 4 characters): road
Enter the text: UENFQRIQ
Decrypted Text: SENTENCE
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3

```

Program 4: Vigenère Cipher

Pseudocode/Algorithm

- **Start:** Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice:** Take the user's choice for encryption, decryption, or exit.
- **Exit Check:** If the choice is to exit, terminate the program.
- **Input Text and Key:** Prompt the user to input the text and the key for encryption or decryption.
- **Generate Key:** Extend the key to match the length of the text, keeping non-alphabetic characters in place.
- **Iterate Over Text:** Loop through each character in the text.
- **Check Alphabetic Characters:** If the character is alphabetic, calculate its new position based on the key and mode (encrypt or decrypt).
- **Handle Non-Alphabetic Characters:** If the character is not alphabetic, retain it in the result.
- **Build Result:** Combine the processed characters to form the encrypted or decrypted text.
- **Display Result:** Print the resulting ciphertext or plaintext and return to step 1 unless the user chooses to exit.

Code

```
def generate_key(text, key):

    key=key.upper()

    ext_key=""

    key_index=0

    for char in text:

        if char.isalpha():

            ext_key+=key[key_index%len(key)]

            key_index+=1

        else:

            ext_key+=char
```

```
    return ext_key

def vigenere(text, key, chk):
    result=""
    key=generate_key(text, key)

    for i, char in enumerate(text):
        if char.isalpha():
            base=ord('A') if char.isupper() else ord('a')
            key_shift=ord(key[i].upper())-ord('A')
            if chk=="e":
                result+=chr((ord(char)-base+key_shift)%26+base)
            else:
                result+=chr((ord(char)-base-key_shift)%26+base)
        else:
            result+=char

    return result

print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
    print("--VIGENERE ENCRYPTION--")
    print("1) Encrypt Text")
    print("2) Decrypt Text")
    print("3) Exit")
    ch=int(input("Enter Choice: "))
```

```

if ch==3:

    break

text = input("Enter the text: ")

key = input("Enter the key: ")

if ch==1:

    print(f"Encrypted: {vigenere(text,key,"e")}")

elif ch==2:

    print(f"Decrypted: {vigenere(text,key,"d")}")

```

Code Screenshot

```

1  def generate_key(text, key):
2      key=key.upper()
3      ext_key=""
4      key_index=0
5
6      for char in text:
7          if char.isalpha():
8              ext_key+=key[key_index%len(key)]
9              key_index+=1
10         else:
11             ext_key+=char
12     return ext_key
13
14 def vigenere(text, key, chk):
15     result=""
16     key=generate_key(text, key)
17
18     for i, char in enumerate(text):
19         if char.isalpha():
20             base=ord('A') if char.isupper() else ord('a')
21             key_shift=ord(key[i].upper())-ord('A')
22             if chk=="e":
23                 result+=chr((ord(char)-base+key_shift)%26+base)
24             else:
25                 result+=chr((ord(char)-base-key_shift)%26+base)
26         else:
27             result+=char
28
29     return result
30
31 print("Siddhanth Monnappa\t22BCE3061\n")
32 while(True):
33     print("--VIGENERE ENCRYPTION--")
34     print("1) Encrypt Text")
35     print("2) Decrypt Text")
36     print("3) Exit")
37     ch=int(input("Enter Choice: "))
38     if ch==3:
39         break
40     text = input("Enter the text: ")
41     key = input("Enter the key: ")
42
43     if ch==1:
44         print(f"Encrypted: {vigenere(text,key,"e")}")
45     elif ch==2:
46         print(f"Decrypted: {vigenere(text,key,"d")}")
47

```

Output

```

Siddhanth Monnappa      22BCE3061

--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter the text: Sentence
Enter the key: round
Encrypted: Jshgheqy
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter the text: Jshgheqy
Enter the key: round
Decrypted: Sentence
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter the text: Protein
Enter the key: Football
Encrypted: Ufcmfiy
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter the text: Ufcmfiy
Enter the key: Football
Decrypted: ProteIn
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3

```