COMPUTER SECURITY LAB WORK

2023

CRYPTOGRAPHY

GROUP MEMBERS

1. Elizabeth Nduta C026-01-0693/2020
2. Paul Maigwa C026-01-1793/2020
3. David Nduati C026-01-1274/2019
4. Brian Minjire C026-01-1339/2019
5. James Njuguna C026-01-0678/2020
6. Haron Mburu C026-01-0734/2020
7. Joseph Muya C026-01-0670/2020
8. Brian Muinde C026-01-0698/2020
9. Evans Musoti C026-01-0745/2020
10. Philip Mucheru C026-01-0671/2020

Columnar Transposition cipher

A Columnar Transposition cipher is a type of transposition cipher, which rearranges the order of the letters in a message to create the ciphertext. The message is written in a grid of rows and columns and then read out column by column in a scrambled order defined by a keyword.

Here's a brief explanation of how it works:

Encryption Process (cipher encryption function):

i. Takes user input for the plain text and a keyword.
ii. Converts the plain text to uppercase and removes spaces.
iii. Assign numbers to the keyword based on alphabetical order.
iv. Prints the keyword and corresponding numbers.
v. Appends extra characters (dots) to the plain text to fit the grid.
vi. Converts the plain text into a grid based on the keyword.
vii. Prints the grid.
viii. Gets the locations of the numbers in the keyword.
ix. Performs columnar transposition to create the cipher text.
x. Prints the resulting cipher text.

Helper Functions:

a) **get_number_location**: Takes a keyword and its assigned numbers, and returns a string representing the positions of the numbers.
b) **keyword_num_assign**: Assign numbers to the keywords' characters based on their alphabet order.

Decryption Process:

i. Takes user input for the cipher text and a keyword.
ii. Converts the cipher text to uppercase and removes spaces.
iii. Assign numbers to the keyword based on alphabetical order.
iv. Gets the locations of the numbers in the keyword.
v. Determines the number of rows in the grid.
vi. Converts the cipher text into a grid based on the keyword and number of locations.
vii. Performs the reverse columnar transposition to obtain the plain text.
viii. Prints the resulting plain text.

Main Function (main function):

i. Takes user input for encryption or decryption.
ii. Calls the corresponding function (cipher_encryption or cipher_decryption).
iii. Prints an error message for an invalid choice.

IMPLEMENTATION WITH PYTHON CODE

ENCRYPTION AND DECRYPTION

```
def cipher_encryption():
    # User input for plain text and keyword
    msg = input("Enter Plain Text: ").replace(" ", "").upper()
    # print(msg)
    key = input("Enter keyword: ").upper()

    # assigning numbers to keywords
    kywrd_num_list = keyword_num_assign(key)

    # printing key
    for i in range(len(key)):
        print(key[i], end=" ", flush=True)
    # for
    print()
    for i in range(len(key)):
        print(str(kywrd_num_list[i]), end=" ", flush=True)
    # for
    print()
    print("-------------------------")
    # in case characters don't fit the entire grid perfectly.
    extra_letters = len(msg) % len(key)
    # print(extraLetters)
    dummy_characters = len(key) - extra_letters
    # print(dummyCharacters)
    if extra_letters != 0:
        for i in range(dummy_characters):
            msg += "."
    # if
            print(msg)
    num_of_rows = int(len(msg) / len(key))

    # Converting message into a grid
    arr = [[0] * len(key) for i in range(num_of_rows)]
    z = 0
    for i in range(num_of_rows):
        for j in range(len(key)):
            arr[i][j] = msg[z]
            z += 1
        # for
    # for
    for i in range(num_of_rows):
        for j in range(len(key)):
            print(arr[i][j], end=" ", flush=True)
        print()
    # for
    # Getting locations of numbers
    num_loc = get_number_location(key, kywrd_num_list)
```

```
    print(num_loc)
    # cipher
    cipher_text = ""
    k = 0
    for i in range(num_of_rows):
        if k == len(key):
            break
        else:
            d = int(num_loc[k])
        # if
        for j in range(num_of_rows):
            cipher_text += arr[j][d]
        # for
        k += 1
    # for
    print("Cipher Text: {}".format(cipher_text))
def get_number_location(key, kywrd_num_list):
    num_loc = ""
    for i in range(len(key) + 1):
        for j in range(len(key)):
            if kywrd_num_list[j] == i:
                num_loc += str(j)
            # if
        # for
    # for
    return num_loc
def keyword_num_assign(key):
    alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    kywrd_num_list = list(range(len(key)))
    # print(kywrdNumList)
    init = 0
    for i in range(len(alpha)):
        for j in range(len(key)):
            if alpha[i] == key[j]:
                init += 1
                kywrd_num_list[j] = init
    return kywrd_num_list
def cipher_decryption():
    msg = input("Enter Cipher Text: ").replace(" ", "").upper()
    # print(msg)
    key = input("Enter keyword: ").upper()
    # assigning numbers to keywords
    kywrd_num_list = keyword_num_assign(key)
    num_of_rows = int(len(msg) / len(key))
    # getting locations of numbers
    num_loc = get_number_location(key, kywrd_num_list)
    # Converting message into a grid
    arr = [[0] * len(key) for i in range(num_of_rows)]
    # decipher
    plain_text = ""
    k = 0
    itr = 0
    # print(arr[6][4])
    # itr = len(msg)
    for i in range(len(msg)):
```

```
        d = 0
        if k == len(key):
            k = 0
        else:
            d: int = int(num_loc[k])
        for j in range(num_of_rows):
            arr[j][d] = msg[itr]
            # print("j: {} d: {} m: {} l: {} ". format(j, d, msg[l], l))
            itr += 1
        if itr == len(msg):
            break
        k += 1
    print()


    for i in range(num_of_rows):
        for j in range(len(key)):
            plain_text += str(arr[i][j])
        # for
    print("Plain Text: " + plain_text)
def main():
    choice = int(input("1. Encryption\n2. Decryption\nChoose(1,2): "))
    if choice == 1:
        print("Encryption")
        cipher_encryption()
    elif choice == 2:
        print("Decryption")
        cipher_decryption()
    else:
        print("Invalid Choice")
if __name__ == "__main__":
    main()
```

OUTPUT

```
==================== RESTART: C:/Users/USER/col trans2.py ================
1. Encryption
2. Decryption
Choose(1,2): 1
Encryption
Enter Plain Text: DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY UNIVERSITY NYERI
Enter keyword: JOSEPH
J O S E P H
3 4 6 1 5 2
-------------------------
D E D A N K
I M A T H I
U N I V E R
S I T Y O F
T E C H N O
L O G Y U N
I V E R S I
T Y N Y E R
I . . . . .
350142
Cipher Text: ATVYHYRY.KIRFONIR.DIUSTLITIEMNIEOVY.NHEONUSE.DAITCGEN.

==================== RESTART: C:/Users/USER/col trans2.py ================
===
1. Encryption
2. Decryption
Choose(1,2): 2
Decryption
Enter Cipher Text: ATVYHYRY.KIRFONIR.DIUSTLITIEMNIEOVY.NHEONUSE.DAITCGEN.
Enter keyword: JOSEPH

Plain Text: DEDANKIMATHIUNIVERSITYOFTECHNOLOGYUNIVERSITYNYERI.....
```