ISSN: 2769-545X

Research Article

Advances in Machine Learning & Artificial Intelligence

Encryption and Decryption Algorithm with The Ability of Different Computing Layers to Send and Receive Data in The Network (Encryption Decryption Multi-Layer Algorithm)

Ahmad Karami Bukani*

Department of Education - Bukan City - West Azarbaijan (Mukriyan Kurdistan) Province – Iran

[*]Corresponding Author
Ahmad Karami Bukani, Department of Education - Bukan City - West Azarbaijan (Mukriyan Kurdistan) Province – Iran

Submitted: 2024 Feb 03; **Accepted**: 2024 Feb 21; **Published**: 2024 Mar 05

Citation: Bukani, A., K. (2024). Encryption and decryption algorithm with the ability of different computing layers to send and receive data in the network (Encryption Decryption Multi-Layer Algorithm). *Adv Mach Lear Art Inte, 5*(1), 01-12.

Abstract
The encryption Decryption Multi-Layer algorithm is in the form of several layers and each layer performs a specific action, in total, this algorithm has eight layers, four layers are for encryption and four layers are for decryption, and these layers guarantee that the best the method of encryption and decryption of very important data.

Keywords: Hide Data, Hide Layer, Gap Data, Gap Layer, Show Layer, Show Data, Encrypt, Decrypt, Multi-Layer, Algorithm

1. Introduction

There are different ways to encrypt and decrypt the data to protect the data from low-level security to a high level but a new solution has formed in my mind that helps to encrypt and decrypt the data in a certain way, according to the existence of four layers for encryption and four layers for decryption, various operations take place on the data to ensure that when the data is sent or received over the wireless network or the wired network when If the data is eavesdropped or falls into the hands of hackers, it will not be recoverable, even if part of the two-step encryption is recoverable, they will not be able to understand the content of the data, Although it is written in English, misleading and fake information has fallen into the hands of hackers [1,2].

2. Overview

I used the Python 3.12 programming language and described the entire set of commands along with the layers of the EDML algorithm, but the layers of this algorithm include two parts: Although it is written in English, misleading and fake information has fallen into the hands of hackers [3].

Part A- Data encryption consists of four layers.
• Adding useless text data to useful text data (INSERT GAP TEXT LAYER)
• Hiding all useful text data and useless text data and converting all data to useless data (Hide TEXT LAYER)
• Inserting the encryption of all useless textual data in the format of step one (INSERT ENCRYPTION RANDOM LAYER) 4-
• insert re-encryption of all encrypted data in the form of the second step (INSERT ENCRYPTION GAP LAYER)
Part B - Data decryption consists of four layers.
• Delete all encrypted data in the second step for decryption (DELETE DECRYPTION GAP LAYER)
• Deleting all encrypted data of step one to decrypt and convert it to all useless text data (DELETE DECRYPTION RANDOM LAYER)
• Convert all useless data to useful text data and useless text data and display it (SHOW TEXT LAYER)
• Deleting all useless text data and revealing useful text data (DELETE GAP TEXT LAYER)

According to (Figure 1), the data encryption layers deliver data from top to bottom, and the data decryption layers deliver data from bottom to top, and the act of sending data in networks is done by the INSERT ENCRYPTION GAP layer, and the operation of receiving data in the networks is done by the DELETE GAP TEXT layer.

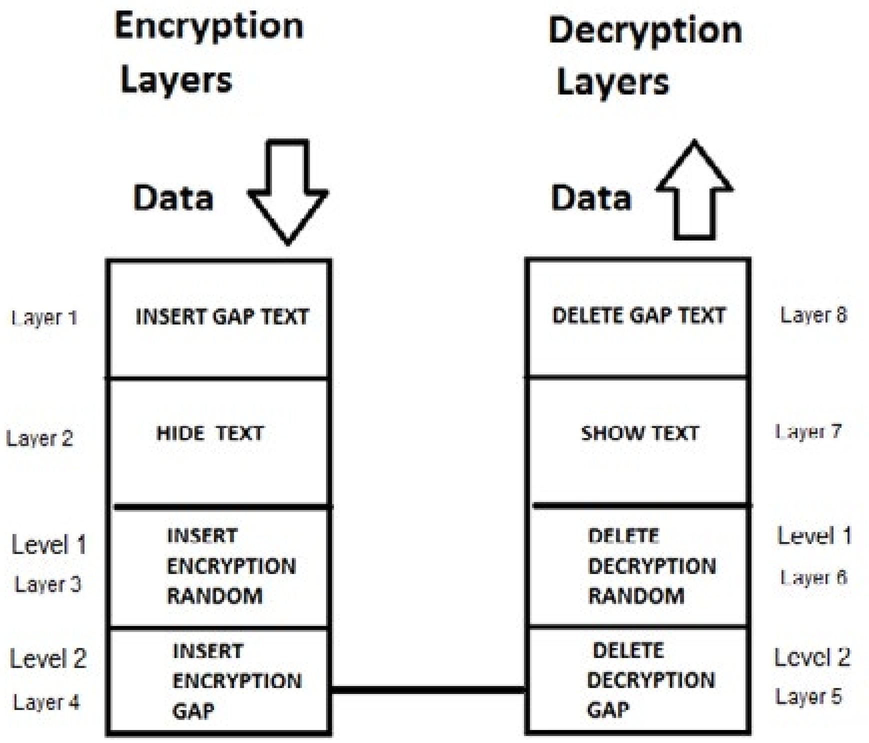


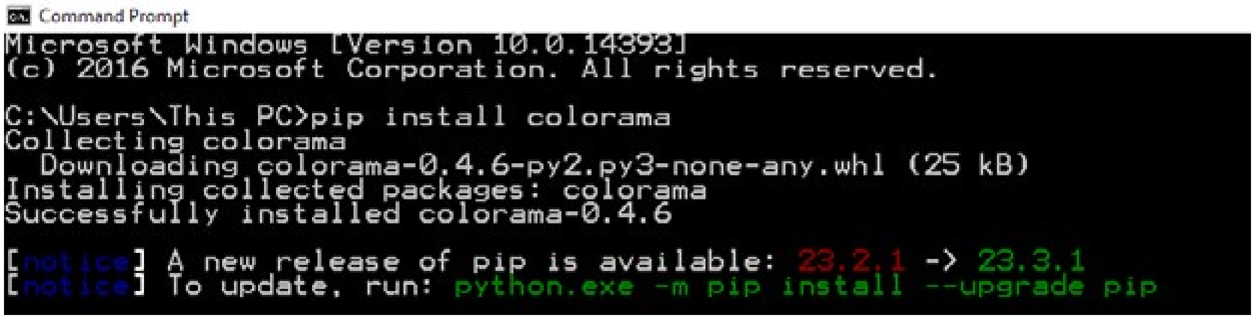
Figure 1: EDML Algorithm

The complete source code has been attached to this article and since some of the code is to show the results better and has no effect on the implementation of the algorithm. To shorten the description of the article, I will omit repetitive codes and codes just for information and graphic displays with various colors. This algorithm has three libraries for generating random numbers, generating colors for displaying texts, and inserting time.

```
import random
import colorama
import time
```

By default, the colorama library, which is used to produce text colors, cannot be recognized and its functions cannot be used, and an error message appears during the output, so the pip command is used to install or update the libraries, according to Figure 2. I have entered its command.

```
pip install colorama
```



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\This PC>pip install colorama
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: colorama
Successfully installed colorama-0.4.6

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Figure 2: Install Library

The original message with the text content to be encrypted is contained in the MyData variable.

```
MyData="I Live in Kurdistan , my+name+is Ahmad.Karami.Bukani , I+am a Programmer , this+is my algorithm ..."
```

The words in the MyData variable are separated from each other and placed in the MyDataList variable, and the sentences are placed in the MyDataList2 variable.

```
MyDataList = []
MyDataList2 = []
```

The position of code words are stored in dictionary variables, based on nouns, verbs, adverbs of place, adverbs of time, etc., words are stored in dictionary variables.

MyDictionaryPosition1_1=[]
MyDictionaryPosition2_1=[]

MyDictionaryPosition1_2=[]
MyDictionaryPosition2_2=[]

MyDictionaryPosition1_3=[]
MyDictionaryPosition2_3=[]

MyDictionaryPosition1_4=[]
MyDictionaryPosition2_4=[]

MyDictionaryPosition1_5=[]
MyDictionaryPosition2_5=[]

MyDictionaryPosition1_6=[]
MyDictionaryPosition2_6=[]

MyDictionaryPosition1_7=[]
MyDictionaryPosition2_7=[]

MyDictionaryPosition1_8=[]
MyDictionaryPosition2_8=[]

MyDictionaryPosition1_9=[]
MyDictionaryPosition2_9=[]

MyDictionaryPosition1_10=[]
MyDictionaryPosition2_10=[]

The ElementCounter variable counts the number of words to be used to encrypt and decrypt data.

ElementCounter=0

Dictionary variables based on nouns, verbs, adverbs of place, adverbs of time, etc. help to convert useful data into useless data and vice versa, and to hide and reveal useful data. Also, each time the algorithm is executed, a list of random words is selected based on nouns, verbs, adverbs of place, adverbs of time, etc.

MyDictionary1 = ["I+am","I+am+not","she+is" , "she+is+not","he+is", "he+is+not"]
MyDictionary2=["this+is","this+is+not","those+are","those+are+not","there+is", "there+is+not"]
MyDictionary3 = ["my+name+is","my+name+is+not"]
MyDictionary4 = ["Live","Love","Enable","Wish","Active","Write","Teaching","Going"]
MyDictionary5 = ["Programmer","Reporter","Worker","Teacher","Driver"]
MyDictionary6 = ["We","I","You","They"]
MyDictionary7=["car","cars","airplane","bus","earth","moon","ship","book","books","note","mobile","notes","notebook","night","day","algorithm"]
MyDictionary8 = ["in","to","from"]
MyDictionary9 = ["Kurdistan","USA","France","UK","Germany","Italy","Norway"]
MyDictionary10=["Ahmad.Karami.Bukani","David","Shaho","Eleya","Mary","Ako","Nishteman","Nechirvan"]

Each time the algorithm is executed, a list of different random numbers is placed in the variable key_random_list and variable key_random_list2 to encrypt and decrypt the data.

key_random_list = []
key_random_list2 = []

Encrypted data is placed in the encrypt variable.

encrypt=""

The decrypted data is placed in the decrypt variable.

decrypt=""

The re-encrypted data is placed in the encrypt_gap variable.

encrypt_gap=""

The re-decrypted data is placed in the decrypt_gap variable.

decrypt_gap=""

The encrypt_gap_send_message variable stores encrypted data and can send this data over the network.

encrypt_gap_send_message=""

The decrypt_correct_receive_message variable stores the decrypted data and this data can be received from the network platform.

decrypt_correct_receive_message=""

The number of encryption and decryption max_len is equal to 20, which can be increased more and more, for example, to 1000, and it will cause long encryption and decryption. The act of deciphering the password becomes extremely difficult, although as the length of the data to encrypt and decrypt the data increases, it is natural that the value of the max_len variable also increases.

max_len=20

The source variable is responsible for the collection of words of useful data and useless data and helps to navigate and encrypt and decrypt the string of useful and useless data.

source=""
The index variable is used to navigate the encrypted and decrypted data.

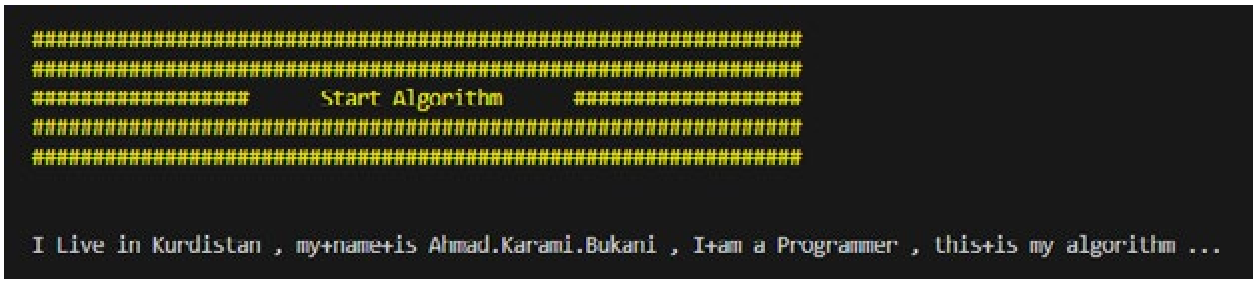
index=0

To delay the execution of the algorithm, the duration variable can be used in seconds, although instead of the time library, the input function can be used to execute the next layers of encryption or decryption.

duration = 5

According to Figure 3, the main data is displayed at the beginning of the algorithm execution with the command print(MyData) and it is the main data that is encrypted and decrypted.

print(MyData)



```
#################################################################
#################################################################
##################      Start Algorithm      ####################
#################################################################
#################################################################


I Live in Kurdistan , my+name+is Ahmad.Karami.Bukani , I+am a Programmer , this+is my algorithm ...
```

Figure 3: Main Data

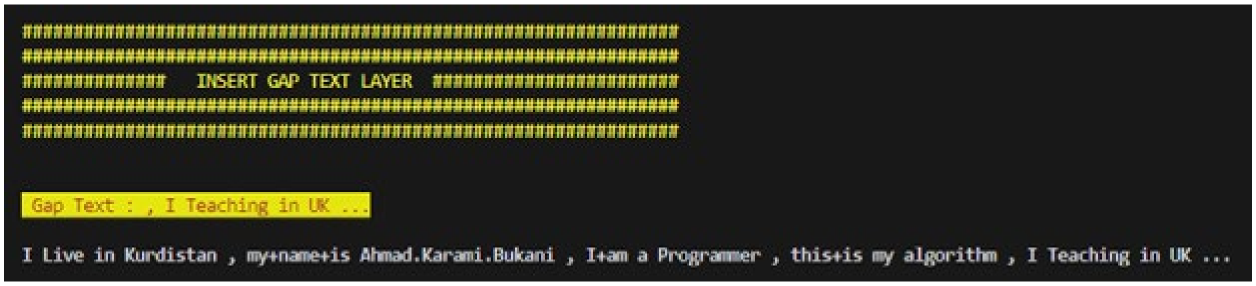
2.1 Data Encryption

As mentioned, the data encryption section is divided into four layers, and each layer performs operations related to encryption, and each layer transfers data from the upper layer to the lower layer.

2.1.1 Insert Gap Text Layer

The first layer adds useless data to useful data, this layer helps to insert and add a lot of useless data to useful data. However here with a simple example and using the GapText variable, the useless data is inserted and added to the useful data.

Using the choice function in the random library, a short sentence is created, this is the useless data in the GapText variable, and it is displayed in the monitor output as shown in (Figure 4).



```
#################################################################
#################################################################
##############   INSERT GAP TEXT LAYER   ########################
#################################################################
#################################################################


Gap Text :  , I Teaching in UK ...

I Live in Kurdistan , my+name+is Ahmad.Karami.Bukani , I+am a Programmer , this+is my algorithm , I Teaching in UK ...
```

Figure 4: Insert Gap Text Layer

GapText = ", "+ random.choice(MyDictionary6)+" "+random.choice(MyDictionary4)+" "+random.choice(MyDictionary8)+" "+random.choice(MyDictionary9)+" ..."

The two print functions set the text color to yellow and the background color to red, mostly displayed for the informative aspect of the GapText variable.

print(colorama.Fore.RED+colorama.Back.YELLOW+colorama.Style.BRIGHT+" Gap Text : "+GapText)

The replace function of the MyData variable helps to combine and add the useless data in the GapText variable with the useful data of the MyData variable, the same operation of inserting the useless data into the useful data is done.

MyData=MyData.replace("...",GapText)

MyData variable displays useless data and useful data in the output using the print function.

Print(MyData)

The split function of the MyData variable separates the words from each other by using the space letter, and the words are placed in the MyDataList variable, which is of the list type.

```
MyDataList=MyData.split(" ")
```

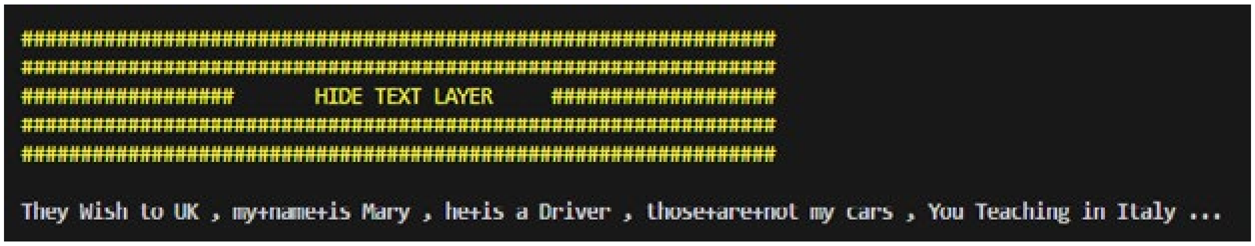
The sleep function from the time library creates a delay using the duration variable, and then after this delay it enters the next stage of the algorithm and the second layer starts to run, in the following layers in the main code of this algorithm from the following code line Used.

```
time.sleep(duration)
```

2.1.2 Hide Text Layer

This layer performs the act of hiding and replacing useless data with useful data, this action helps to protect the data more.

According to (Figure 5), the output of this layer is displayed and the algorithmic implementation of this layer is that the ElementCounter variable counts the words processing useful data and useless data. The first for loop scrolls the list of words, and for each word selected, one unit is added to the ElementCounter variable. The second for loop for the length of the dictionary of the first replacement words, if the word in the list of useful data and useless data is equal to the data list of the first dictionary, a word is randomly selected from the first dictionary using the choice function. The selected word in the list of useful data and useless data and the randomly selected word using the choice function will be placed in the variables of the first position of the dictionary in the choice variable. Then the chosen random word is replaced using the choice variable in the MyDataList variable.



```
##################################################################
##################################################################
##################       HIDE TEXT LAYER      ###################
##################################################################
##################################################################

They Wish to UK , my+name+is Mary , he+is a Driver , those+are+not my cars , You Teaching in Italy ...
```

Figure 5: Hide Text Layer

```
ElementCounter=-1
for element in MyDataList:
 ElementCounter=ElementCounter+1
 for i in range(0,len(MyDictionary1)):
 if element == MyDictionary1[i]:
 choice = random.choice(MyDictionary1)
 MyDictionaryPosition1_1.append(element)
 MyDictionaryPosition2_1.append(choice)
 MyDataList[ElementCounter] = choice
```

As you can see, the next block of code repeats the same routine but instead includes the length of the second dictionary variable and the position variable of the second dictionary. By using variable i, like the previous block, words are replaced in MyDataList variable.

```
ElementCounter=-1
for element in MyDataList:
 ElementCounter=ElementCounter+1
 for i in range(0,len(MyDictionary2)):
 if element == MyDictionary2[i]:
 choice = random.choice(MyDictionary2)
 MyDictionaryPosition1_2.append(element)
 MyDictionaryPosition2_2.append(choice)
 MyDataList[ElementCounter] = choice
```

Next, according to the two loops, the third dictionary variable and the third dictionary position variables also navigate the list of useful data and useless data. The act of replacing nouns, subjects, adverbs of time, adverbs of place, etc., using the ElementCounter variable for the MyDataList variable is performed by the randomly selected word of the choice variable using the choice function. The fourth, fifth, and tenth dictionaries perform the replacement of useful data and useless data, and these replacements cause useful data and useless data to be replaced with completely useless data.

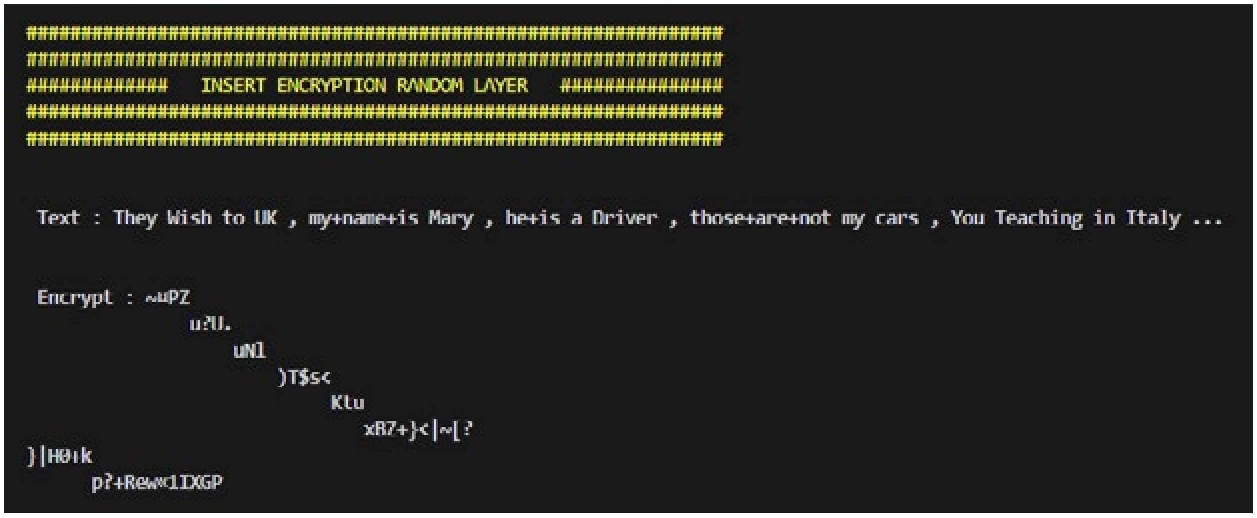
```
ElementCounter=-1
for element in MyDataList:
 ElementCounter=ElementCounter+1
 for i in range(0,len(MyDictionary3)):
 if element == MyDictionary3[i]:
 choice = random.choice(MyDictionary3)
 MyDictionaryPosition1_3.append(element)
 MyDictionaryPosition2_3.append(choice)
 MyDataList[ElementCounter] = choice
```

The for loop for the variable MyDataList is used to show the replacement of useful data and useless data with completely useless data, this layer helps to make useful data completely useless and completely hidden and inaccessible.

```
for element in MyDataList:
 print(element, end=" ")
```

2.1.3 Insert Encryption Random Layer

The third layer encrypts completely useless data and turns it into non-English data, Ascii codes can be used to perform encryption (Level 1 in Figure 1).



```
##########################################################################
##########################################################################
##############    INSERT ENCRYPTION RANDOM LAYER    ###############
##########################################################################
##########################################################################


 Text : They Wish to UK , my+name+is Mary , he+is a Driver , those+are+not my cars , You Teaching in Italy ...


 Encrypt : ~uPZ
                u?U.
                    uNl
                        )T$s<
                             Klu
                                xR7+}<|~[?
}|H0ik
      p?+Rew«1IXGP
```

Figure 6: INSERT ENCRYPTION RANDOM LAYER

Useless data should be converted from words in the MyDataList variable from a list type to a string, so using the source variable, the process of converting the list to a string is done.

```
For element in MyDataList:
 source= source + element + " "
```

After converting and creating string, completely useless data, using the randint function from the random library, a random number is selected between zero and fifty. Due to the existence of the for loop and the value of zero to the maximum length of max_len numbers, random numbers are generated and added to the key_random_list variable.

```
for i in range(0, max_len):
 key_random_list.append(random.randint(0,50))
```

Using the print function, completely useless data strings are displayed in the output.

```
print(" Text : " + source)
```

The for loop helps to sum each character in the completely useless source data string, using the ord function, the ASCII codes of the key_random_list variable with a random number. The sum of these two numbers creates a new random character using the chr function, and each new random character that is encrypted is added and pasted with the encrypt variable. If index, which is the length of the string, is equal to the length of the maximum random number max_len, index will be zero again and the list of random numbers will be used again. If the length of the encrypted string is equal to the index variable using the len function, it means that the entire string of useless data has been scrolled and encrypted, and it is exited from the for loop. Of course, every time a character from the completely useless string is checked, one unit is added to the index and the initial value of the index variable is zero.

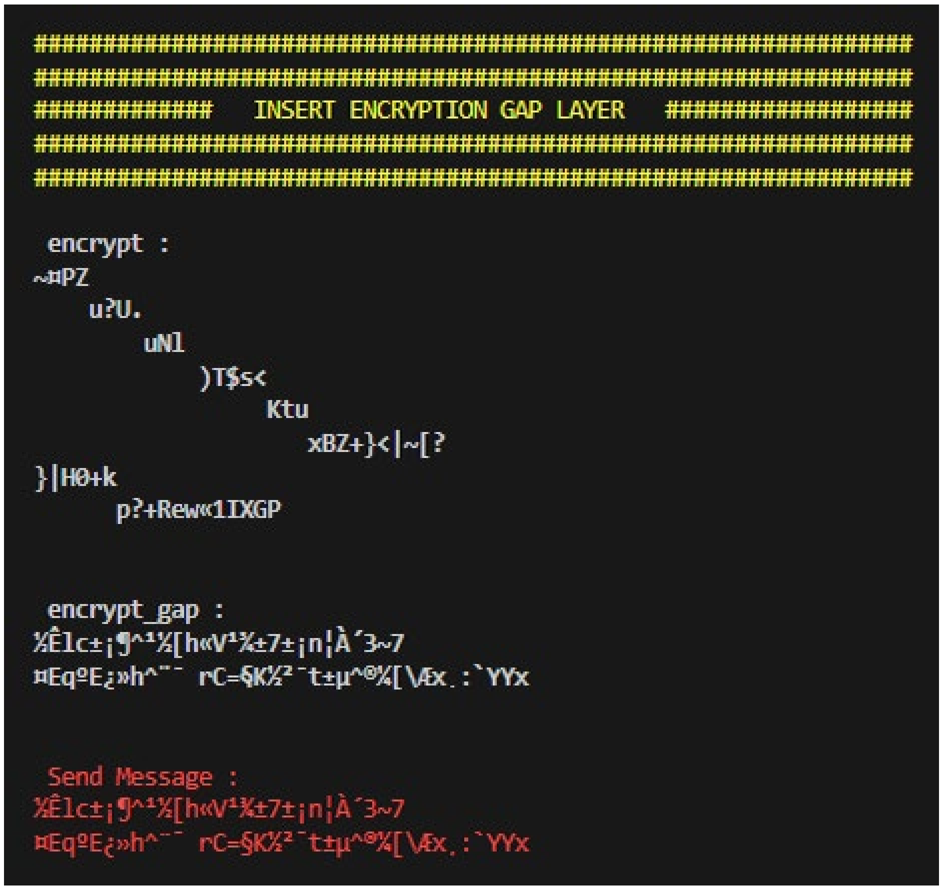
```
index=0
For char in source:
 if(index==max_len):
 index=0
 encrypt+=chr(ord(char)+key_random_list[index])
 if(index==len(source)):
 break
 index+=1
```

At the end of this layer, according to (Figure 6), using the print function, the encrypted string value is displayed in the output in the format of step one.

```
print(" Encrypt : " + encrypt)
```

2.1.4 Insert Encryption Gap Layer

This layer helps to encrypt more and more useless data, the more useless data is encrypted, the harder and harder it is to discover useful data (Level 2 in Figure 1).



```
################################################################
################################################################
#############   INSERT ENCRYPTION GAP LAYER   ##################
################################################################
################################################################

 encrypt :
~¤PZ
    u?U.
        uN1
            )T$s<
                 Ktu
                    xBZ+}<|~[?
}|H0+k
      p?+Rew«1IXGP


 encrypt_gap :
½Êlc±¡¶^¹½[h«V¹¾±7±¡n¦À´3~7
¤EqºE¿»h^¨¯ rC=§K½²¯t±µ^®¼[\Æx¸:`YYx


 Send Message :
½Êlc±¡¶^¹½[h«V¹¾±7±¡n¦À´3~7
¤EqºE¿»h^¨¯ rC=§K½²¯t±µ^®¼[\Æx¸:`YYx
```

Figure 7: Insert Encryption Gap Layer

Using the print function, the encrypted string value is displayed in the output in the format of step one.

```
print(" encrypt : ")
print(encrypt)
```

The encrypt_gap variable can be initialized empty or with some junk data. Of course, if initialization is done with junk data, the junk data will be shown next to the encrypted data in the format of step one. My preference is to use the encrypt_gap variable with an empty value.

```
#encrypt_gap=" 1234567890 "
encrypt_gap=""
```

Again, using the randint function from the random library, a random number is selected between the numbers zero to fifty, and due to the existence of the for loop and the value of zero to the maximum length of max_len numbers, random numbers are generated and added to the key_random_list2 variable.

```
for i in range(0,max_len):
 key_random_list2.append(random.randint(0,50))
```

Again, the for loop helps that every single character in the encrypted data string is added in the format of the first step of encrypt, using the ord function, the ASCII codes of the variable key_random_list2 with a random number. The sum of these two numbers creates a new random character using the chr function, and each new random character that is re-encrypted is appended with the encrypt_gap variable. If index, which is the length of the string, is equal to the length of the maximum random number max_len, the value of index will be zero and the list of random numbers will be used again. If, using the len function, the length of the encrypted string is equal to the index variable, that means the entire encrypted data string of step one has been scrolled and re-encrypted and exits the for loop. Of course, every time a character from the coded string is checked in step one format, one unit is added to the index and the initial value of the index variable is zero.

```
index=0
for char in encrypt:
 if(index==max_len):
 index=0
 encrypt_gap+=chr(ord(char)+key_random_list2[index])
 if(index==len(encrypt_gap)):
 break
 index+=1
```

Using the print function, the value of the encrypted string of the second step is displayed in the output.

```
print(" encrypt_gap : ")
print(encrypt_gap)
```

The amount of encrypted data in the format of the second step in the encrypt_gap variable is set into the encrypt_gap_send_message variable.

```
encrypt_gap_send_message = encrypt_gap
```

Finally, according to (Figure 7) in this layer, the amount of encrypted data can be sent in the format of the second step, in the network platform, and the sending message is displayed in red. The encrypt_gap_send_message variable is displayed with the encrypted data in the second step format.

```
print(colorama.Fore.RED+ " Send Message : ")
print(encrypt_gap_send_message)
```

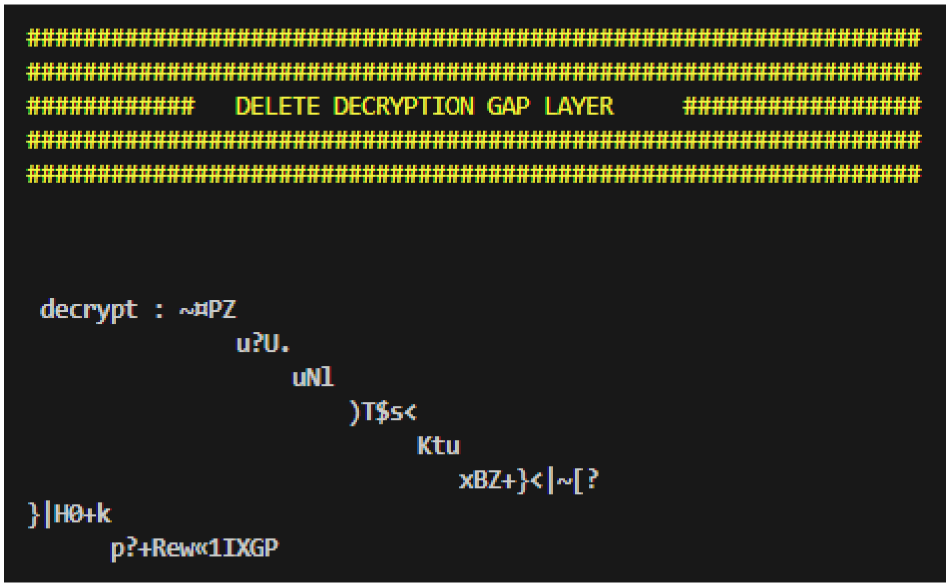
2.2 Data Decryption

The data decryption section is divided into four layers, each layer performs decryption operations, and each layer transfers data from the lower layer to the upper layer.

2.2.1 Delete Decryption Gap Layer

When this layer receives the encryption data string in the stage two format, it converts the encryption data string in the stage two format to the encryption data string in the stage one format and the encryption data string in the stage one format that remains, then it is checked in the next layer. Because the data is encrypted in two layers, then the data is also decrypted in two layers. In the next two layers, the encryption data string is converted into completely useless data and then completely useless data into useful data and useless data. Finally, in the last layer, the useful data is revealed among the useful data and the useless data, and all the useless data is removed and the useful data is displayed (Level 2 in Figure 1).

In this layer, as you can see in (Figure 8), by using the index variable, the act of decryption of the encrypted data string is performed in the second step format. The for loop helps to subtract each character in the encrypted data string in the format of the second step in the encrypt_gap variable, using the ord function, the Asciid codes of the 2key_random_list variable with a random number. By subtracting these two numbers and using the chr function, each character is decrypted and the encrypted data string is attached and added to the decrypt variable in the format of step one. If the index, which is the length of the string, is equal to the length of the maximum random number max_len, the index will be zero again and the list of random numbers will be used again, and if using the len function, the length of the coded string in the second step format is equal to the index variable. It means that the entire encrypted string has been traversed and the second stage of decryption has been completed. Then the for loop ends, of course, every time a character from the encrypted string is checked in the second step format, one unit is added to the index, but the initial value of the index is zero.



```
################################################################
################################################################
############   DELETE DECRYPTION GAP LAYER     #################
################################################################
################################################################



 decrypt : ~¤PZ
               u?U.
                   uNl
                       )T$s<
                            Ktu
                               xBZ+}<|~[?
}|H0+k
      p?+Rew«1IXGP
```

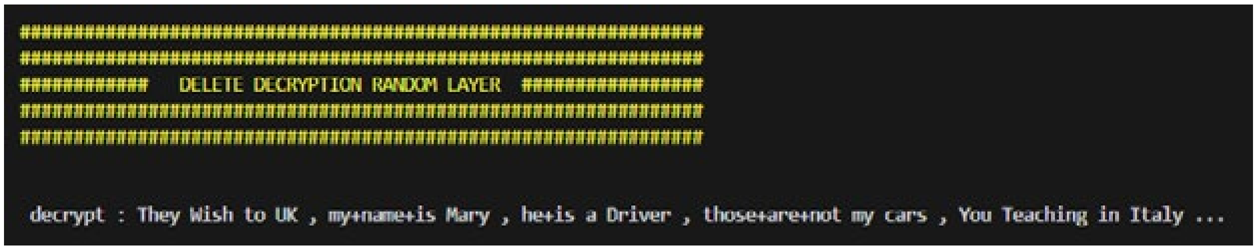
Figure 8: Delete Decryption Gap Layer

```
decrypt=""
index=0
for char in encrypt_gap:
 if index==max_len:
index=0
decrypt+=chr(ord(char)-key_random_list2[index])
if index==len(encrypt_gap):
break
```

```
 index+=1

print(" decrypt : " + decrypt)
```

2.2.2 Delete Decryption Random Layer

To delete and decrypt in the first step format, using this layer, you can get completely useless data and completely decrypt the encrypted data string in the first step format (Level 1 in Figure 1).



```
##################################################################
##################################################################
############   DELETE DECRYPTION RANDOM LAYER   ###################
##################################################################
##################################################################


 decrypt : They Wish to UK , my+name+is Mary , he+is a Driver , those+are+not my cars , You Teaching in Italy ...
```

Figure 9: Delete Decryption Random Layer

By using the index variable, the process of decryptioning the encrypted data string is done in the format of step one. The for loop helps to subtract each character in the encrypted data string in the format of step one in the decrypt variable, using the ord function, the Asciid codes of the key_random_list variable with a random number. By subtracting these two numbers and using the chr function, each character is decrypted and appended with the decrypt_gap variable, and the completely useless data string is recovered. If the index, which is the length of the string, is equal to the length of the maximum random number max_len, the index will be zero again and the list of random numbers will be used again, and if the length of the coded string in the step one format is equal to the index variable using the len function, That is, the entire string encrypted in the form of step one has been scrolled and decrypted completely, and then the completely useless data string is retrieved and then the for loop ends. Of course, every time a character from the encrypted string is checked in the first step format, a unit is added to the index, but the initial value of the index is zero.

```
decrypt_gap=""
index=0
for char in decrypt:
 if index==max_len:
 index=0
 decrypt_gap+=chr(ord(char)-key_random_list[index])
 if index==len(decrypt):
 break
 index+=1
```

In (Figure 9) the result of this layer is displayed and using the print function, it displays the decrypted string that has been converted into completely useless data in the output of the monitor.

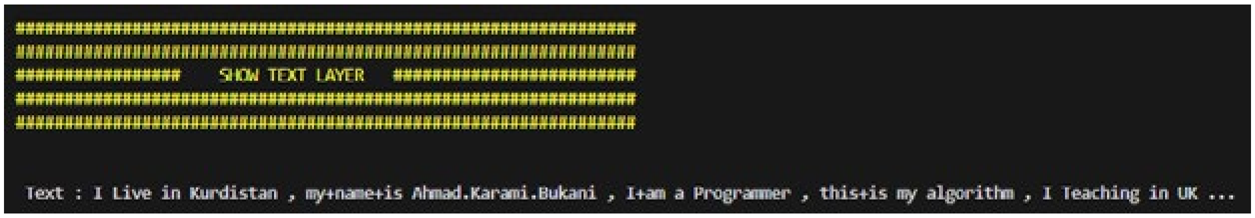
```
print(" decrypt : " + decrypt_gap)
```

The content of the decrypt_gap variable, which contains completely useless data, is assigned to the decrypt variable.

```
decrypt = decrypt_gap
```

2.2.3 Show Text Layer

In this layer, completely useless data is converted into useful data and useless data and helps to maintain relative protection for the data.



```
##################################################################
##################################################################
################    SHOW TEXT LAYER    ###########################
##################################################################
##################################################################


 Text : I Live in Kurdistan , my+name+is Ahmad.Karami.Bukani , I+am a Programmer , this+is my algorithm , I Teaching in UK ...
```

Figure 10: Show Text Layer

The content of the decrypt variable, which contains completely useless data, is set in the MyData variable.

```
MyData=decrypt
```

The content of the MyData variable contains completely useless data, but using the split function, the set of words will be placed in the MyDataList variable and will contain a set of words.

```
MyDataList=MyData.split(" ")
```

The ElementCounter variable counts words while processing completely useless data, the first for loop scrolls the list of MyDataList words, and one unit is added to the ElementCounter variable for each word selected. The second for loop for the length of the dictionary of the first replacement words, if the word in the useless data list is equal to the data list of the first dictionary and the word count using the ElementCounter variable is less than the length of the MyDataList variable of the list type. Using word quantification, the useful data is selected from the first dictionary and the selected word is placed in the choice variable in the list of useful data, Then, by using the choice variable, the action of replacing the useless data is done, and in fact, the useful data has been replaced.

```
ElementCounter=-1
for element in MyDataList:
 ElementCounter=ElementCounter+1
 for i in range(0,len(MyDictionaryPosition2_1)):
     if element == MyDictionaryPosition2_1[i] and ElementCounter<len(MyDataList):
 choice = MyDictionaryPosition1_1[i]
 MyDataList[ElementCounter] = choice
 break
```

In the continuation of the codes of this layer, the next block of codes is repeated with the same procedure, but instead of that, the length of the second dictionary variable and the position variable of the second dictionary are taken into account, and using the i variable, the words are replaced in the MyDataList variable, as in the previous block. Each time after the replacement, the inner for loop is broken and the jump to the outer for loop is performed.

```
ElementCounter=-1
for element in MyDataList:
 ElementCounter=ElementCounter+1
 for i in range(0,len(MyDictionaryPosition2_2)):
     if element == MyDictionaryPosition2_2[i] and ElementCounter<len(MyDataList):
 choice = MyDictionaryPosition1_2[i]
 MyDataList[ElementCounter] = choice
 break
```

It is clear and clear that according to the two loops, the third dictionary variable and the third variables of the dictionary position also navigate the list of useful data and useless data, and the action of replacing nouns, subjects, adverbs of time, adverbs of place, etc., using the ElementCounter variable for The MyDataList variable is handled by the choice variable. The next dictionaries, fourth, fifth, and up to the tenth dictionaries perform the replacement of useless data, and these replacements make completely useless data become useful data and useless data.

```
ElementCounter=-1
for element in MyDataList:
 ElementCounter=ElementCounter+1
 for i in range(0,len(MyDictionaryPosition2_3)):
     if element == MyDictionaryPosition2_3[i] and ElementCounter<len(MyDataList):
 choice = MyDictionaryPosition1_3[i]
 MyDataList[ElementCounter] = choice
 break
```

The for loop navigation for the variable MyDataList is to show the conversion of useful data and useless data, and in this way, the string of useful data and useless data is placed in the source variable.

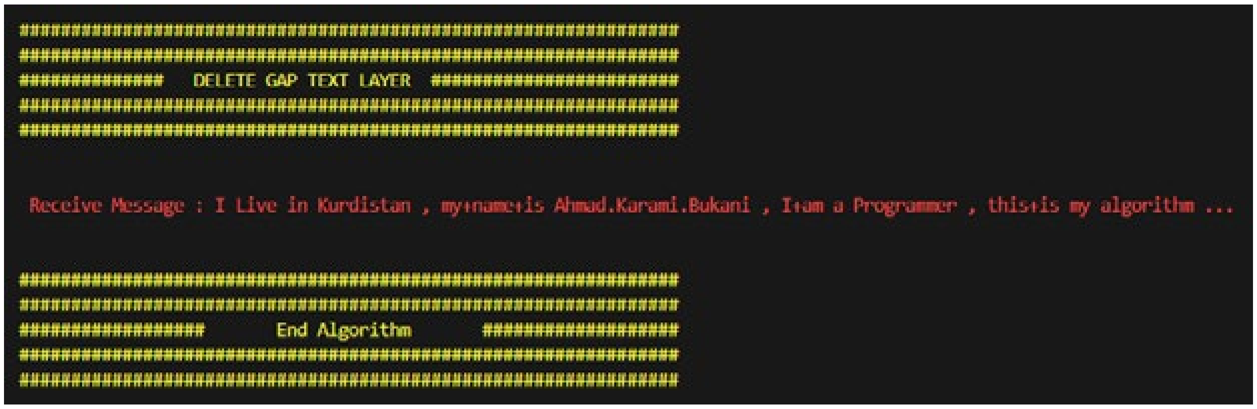
```
source=""
for element in MyDataList:
 source = source + element + " "
```

According to Figure 10, finally, it is displayed in the source variable, which contains the string of useful data and useless data, using the print function.

```
print(" Text : " + source)
```

2.2.4 Delete Gap Text Layer

In this layer, useful data and useless data are converted into completely useful data and the algorithm ends.



```
####################################################################
####################################################################
###############   DELETE GAP TEXT LAYER  ###########################
####################################################################
####################################################################


 Receive Message : I Live in Kurdistan , my+name+is Ahmad.Karami.Bukani , I+am a Programmer , this+is my algorithm ...


####################################################################
####################################################################
###################       End Algorithm      #######################
####################################################################
####################################################################
```

Figure 11: DELETE GAP TEXT LAYER

Using the split function from the source variable, the parts of the sentences are separated using the character and the list of these sentences is placed in the MyDataList2 variable.

```
MyDataList2=source.split(",")
```

The source variable takes a null value.

source=""

By using the source variable and the length of the MyDataList2 variable and by traversing the MyDataList2 variable using the for loop, the sentences are combined and pasted together. The last sentence, which is considered useless data, will be removed from the useful data, and the source variable will contain only useful data.

```
for i in range(0,len(MyDataList2)-1):
 source = source + MyDataList2[i] + ","
```

By using the \b character, a letter at the end of the sentence, i.e. the character, is removed, and then the characters ... are added to the end of the sentence, quite useful data.

```
source = source +"\b"
source = source +"..."
```

And the source variable is set in the decrypt_correct_receive_message variable and the data decryption is done completely.

```
decrypt_correct_receive_message=source
```

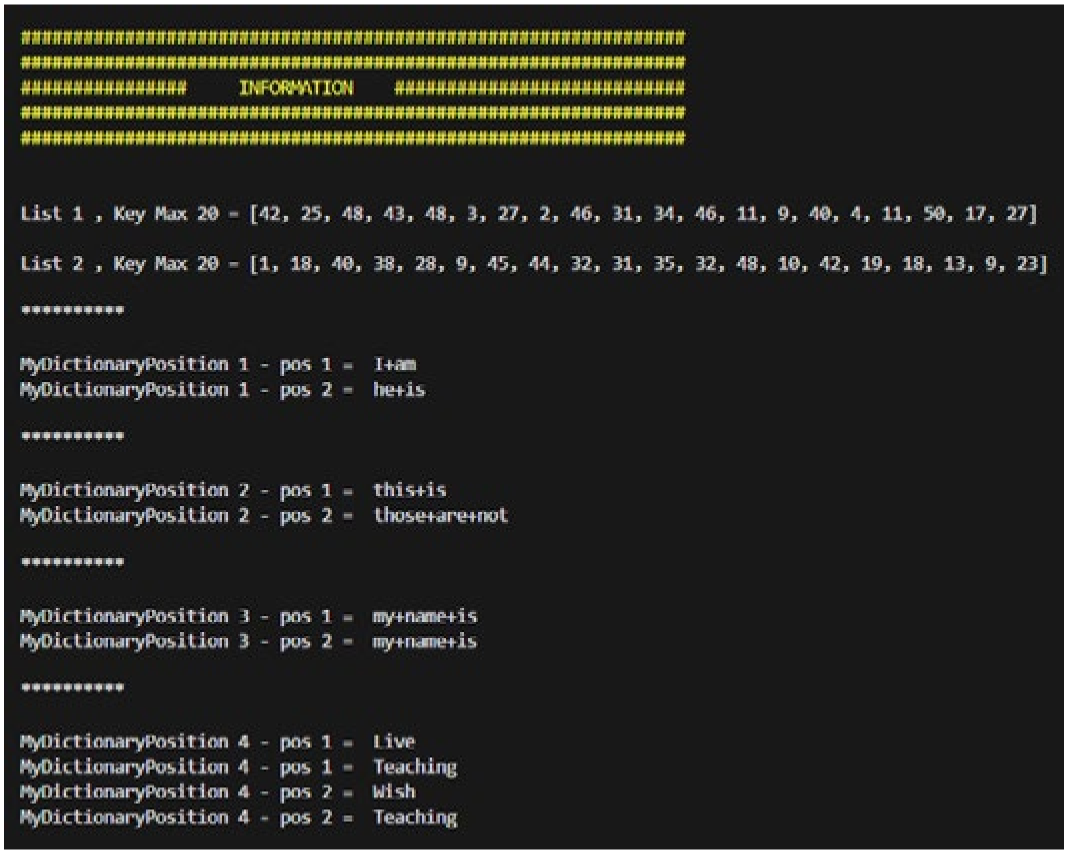
The amount of completely useful data can be received in the network platform and the message of the variable decrypt_correct_receive_message is displayed in red.

```
print(colorama.Fore.RED+" Receive Message : "+ decrypt_correct_receive_message)
```

The algorithm ends here and all the steps are done successfully, the output of this layer is shown in (Figure 11).

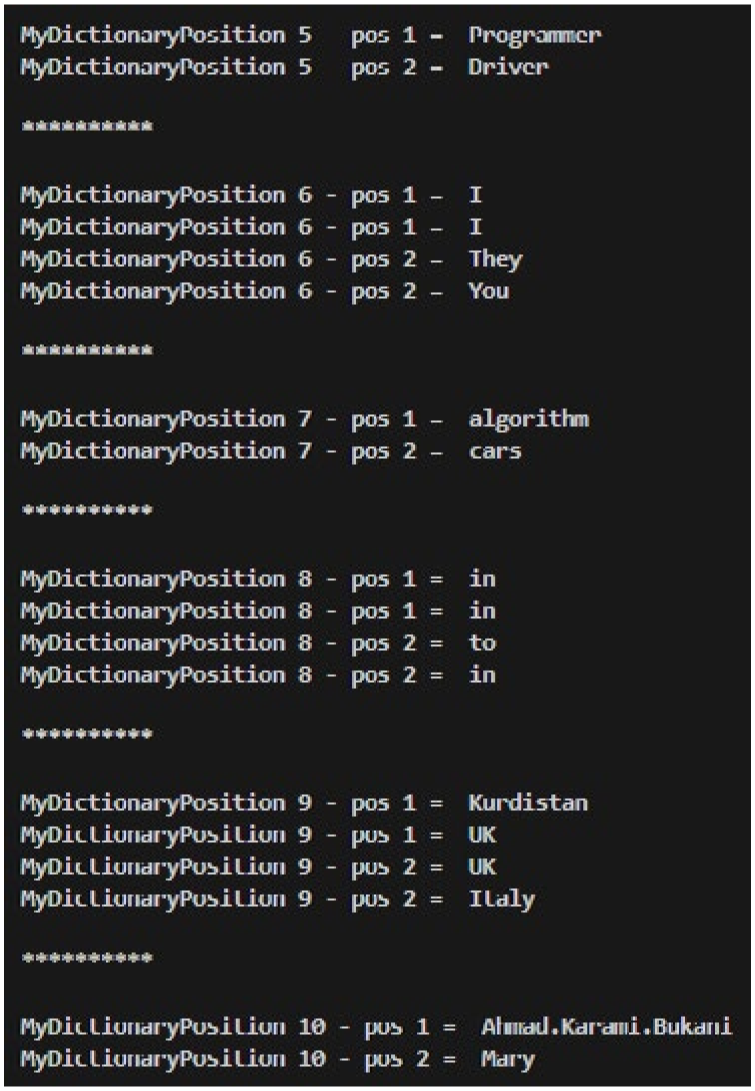
2.3 Keys List

As you can see in Part A and Part B in Figure 12, the list of random keys for data encryption and decryption as well as the position list and the list of useful words and useless words are displayed, and with this algorithm, the success of encryption and the success of decryption It is guaranteed.



```
##########################################################################
##########################################################################
#################     INFORMATION     ####################################
##########################################################################
##########################################################################


List 1 , Key Max 20 = [42, 25, 48, 43, 48, 3, 27, 2, 46, 31, 34, 46, 11, 9, 40, 4, 11, 50, 17, 27]

List 2 , Key Max 20 = [1, 18, 40, 38, 28, 9, 45, 44, 32, 31, 35, 32, 48, 10, 42, 19, 18, 13, 9, 23]

**********

MyDictionaryPosition 1 - pos 1 =  I+am
MyDictionaryPosition 1 - pos 2 =  he+is

**********

MyDictionaryPosition 2 - pos 1 =  this+is
MyDictionaryPosition 2 - pos 2 =  those+are+not

**********

MyDictionaryPosition 3 - pos 1 =  my+name+is
MyDictionaryPosition 3 - pos 2 =  my+name+is

**********

MyDictionaryPosition 4 - pos 1 =  Live
MyDictionaryPosition 4 - pos 1 =  Teaching
MyDictionaryPosition 4 - pos 2 =  Wish
MyDictionaryPosition 4 - pos 2 =  Teaching
```

Part: A



```
MyDictionaryPosition 5   pos 1 -  Programmer
MyDictionaryPosition 5   pos 2 -  Driver

**********

MyDictionaryPosition 6 - pos 1 -  I
MyDictionaryPosition 6 - pos 1 -  I
MyDictionaryPosition 6 - pos 2 -  They
MyDictionaryPosition 6 - pos 2 -  You

**********

MyDictionaryPosition 7 - pos 1 -  algorithm
MyDictionaryPosition 7 - pos 2 -  cars

**********

MyDictionaryPosition 8 - pos 1 =  in
MyDictionaryPosition 8 - pos 1 =  in
MyDictionaryPosition 8 - pos 2 =  to
MyDictionaryPosition 8 - pos 2 =  in

**********

MyDictionaryPosition 9 - pos 1 =  Kurdistan
MyDictionaryPosition 9 - pos 1 =  UK
MyDictionaryPosition 9 - pos 2 =  UK
MyDictionaryPosition 9 - pos 2 =  Italy

**********

MyDictionaryPosition 10 - pos 1 =  Ahmad.Karami.Bukani
MyDictionaryPosition 10 - pos 2 =  Mary
```

Part: B

Figure 12: Keys List for Encryption & Decryption

3. Conclusions

Although this algorithm has memory overheads and occupies additional memory for more data, this algorithm guarantees that the data is encrypted or decrypted in different layers, and this encryption and decryption is done in depth.

This algorithm is an example that can be extended and more complex or complete versions published by others in the future, this algorithm guarantees that if the encrypted text information or the decrypted text information falls into the hands of hackers or subversive or terrorist organizations, they will not be able to access correct information, and if hackers access, misleading textual data or misleading encryption or misleading decryption will fall into the hands of subversive organizations or terrorists.

To access the output code and video of this algorithm, you can refer to the appendix of the article.

References

1. Tanenbaum, A. S. (2003). *Computer networks*. Pearson Education India.
2. Neapolitan, R. (2003). Fundamentals of Algorithms Using C++ Pseudocode.
3. Python.