Exercise on Symmetric Encryption Techniques

January 26, 2018

Name: SONGA MUGABE Fabrice
Click HERE to go to the Link of the Code
Click here to Access the Code for Plairfair Algorithm

1 Program for Caesar Cipher Algorithm

```
In [162]: L2I = dict(zip("ABCDEFGHIJKLMNOPQRSTUVWXYZ",range(26)))
          I2L = dict(zip(range(26),"ABCDEFGHIJKLMNOPQRSTUVWXYZ"))

          key = 3
          plaintext = "HELLO"

          # Encryption
          ciphertext = ""
          for c in plaintext.upper():
              if c.isalpha():
                  ciphertext += I2L[ (L2I[c] + key)%26 ]
              else: ciphertext += c

          # Decryption
          plaintext2 = ""
          for c in ciphertext.upper():
              if c.isalpha():
                  plaintext2 += I2L[ (L2I[c] - key)%26 ]
              else: plaintext2 += c

          print ('The Plain Text is:',plaintext)
          print ("The Encrypted Message of ",plaintext," is:",ciphertext)
          print ("The Decrypted Message of",ciphertext," is:",plaintext2)
```

```
('The Plain Text is:', 'HELLO')
('The Encrypted Message of ', 'HELLO', ' is:', 'KHOOR')
('The Decrypted Message of', 'KHOOR', ' is:', 'HELLO')
```

```
In [71]: print L2I
```

```
{'A': 0, 'C': 2, 'B': 1, 'E': 4, 'D': 3, 'G': 6, 'F': 5, 'I': 8, 'H': 7, 'K': 10, 'J': 9, 'M': 12, 'L':
```

```
In [72]: print I2L
```

```
{0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G', 7: 'H', 8: 'I', 9: 'J', 10: 'K', 11: 'L', 12:
```

2 Program for Hill Cipher Scheme Algorithm

```
In [161]: import numpy as np

          plaintext = "Mississippik"
          P = plaintext.upper();
          print ("The Plain Text is:", P)

          K = np.array([[3, 25], [24, 17]])
          print ("The Key Matrix is:", K)

          output = []
          for character in P:
              number = ord(character)%65
              output.append(number)

          output = np.array(output)
          print output

          #Break the Matrix into 2D matrix
          a = output[:2]
          b = output[2:4]
          c = output[4:6]
          d = output[6:8]
          e = output[8:10]
          f = output[10:12]

          A = np.array([a,b,c,d,e,f])
          A = A.T                          #Transpose of the Matrix

          result = [[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
          for i in range(len(K)):
             # iterate through columns of Y
             for j in range(len(A[0])):
                 # iterate through rows of Y
                 for k in range(len(A)):
                      result[i][j] += (K[i][k] * A[k][j]) % 26
                      print result

          result = np.array(result)
          result = result.T
          result

          B = result[0:7]

          C = B.ravel()   #convert the Matrix into 1D array
          print C

          C = C[:12]%26     #Calculating the Modulus of the remaining numbers
          C = np.array(C)
          print C[:12]

          Hillcipher = ""
          for i in range(len(C)):
              Hillcipher += I2L[C[i]]
              print Hillcipher


          print ("The Encrypted Hill Cipher of", P," is:", Hillcipher)
```

```
('The Plain Text is:', 'MISSISSIPPIK')
('The Key Matrix is:', array([[ 3, 25],
       [24, 17]]))
[12  8 18 18  8 18 18  8 15 15  8 10]
[[10, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 2, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 24, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 2, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 0, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 19, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 0], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 24], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [0, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [2, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 0, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 16, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 0, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 10, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 30, 0, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 30, 16, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 30, 22, 0, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 30, 22, 22, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 30, 22, 43, 0]]
[[28, 10, 32, 20, 30, 40], [8, 36, 30, 22, 43, 10]]
[[28, 10, 32, 20, 30, 40], [8, 36, 30, 22, 43, 24]]
[28  8 10 36 32 30 20 22 30 43 40 24]
[ 2  8 10 10  6  4 20 22  4 17 14 24]
C
CI
CIK
CIKK
CIKKG
CIKKGE
CIKKGEU
CIKKGEUW
CIKKGEUWE
CIKKGEUWER
CIKKGEUWERO
CIKKGEUWEROY
('The Encrypted Hill Cipher of', 'MISSISSIPPIK', ' is:', 'CIKKGEUWEROY')
```

3 Program for Rail Fence Cipher Scheme Algorithm

```
In [160]: plaintext = "defendhim"
          key = 3

          P = plaintext.upper();
          print ("The Plain Text is:", P)

          print ("The Key Matrix is:", key)

          k = list(P)

          m = np.reshape(k, (key, -1))
          m = np.array(m)
          c = m.T

          # C = m.ravel()
          # C
          m[:3]
          s = np.array(c).tolist()
          cipher = c.flatten()

          ciher = ""
          for i in range(len(cipher)):
              ciher += cipher[i]
              print ciher


          print ("The Encrypted Rail Fence Cipher of", P, " is:", ciher)
```

```
('The Plain Text is:', 'DEFENDHIM')
('The Key Matrix is:', 3)
D
DE
DEH
DEHE
DEHEN
DEHENI
DEHENIF
DEHENIFD
DEHENIFDM
('The Encrypted Rail Fence Cipher of', 'DEFENDHIM', ' is:', 'DEHENIFDM')
```

4 Program for Playfair Cipher Scheme Algorithm

```
def printFence(fence):
    for rail in range(len(fence)):
        print ''.join(fence[rail])

def encryptFence(plain, rails, offset=0, debug=False):
    cipher = ''

    # offset
    plain = '#'*offset + plain

    length = len(plain)
    fence = [['#']*length for _ in range(rails)]

    # build fence
    rail = 0
    for x in range(length):
        fence[rail][x] = plain[x]
        if rail >= rails-1:
            dr = -1
        elif rail <= 0:
            dr = 1
        rail += dr

    # print pretty fence
    if debug:
        printFence(fence)

    # read fence
    for rail in range(rails):
        for x in range(length):
            if fence[rail][x] != '#':
                cipher += fence[rail][x]
    return cipher

    # print pretty fence
    if debug:
        printFence(fence)

    # read fence
    for i in range(length):
        for rail in range(rails):
            if fence[rail][i] != '#':
                plain += fence[rail][i]
    return plain


if __name__ == "__main__":
    plain = "DEFENDHIM"
    print plain
    cipher = encryptFence(plain, 3, offset=4, debug=True)
    print cipher
```

4.1 Output for Playfair Cipher

```
fafasonga-2:Data_sec admin$ python playfair.py

Playfair Cipher
Choose :
1,Encrypting
2,Decrypting
1
```

```
Please input the key : MONARCHY
Please input the message : HAMMER

Encrypting:
Message: HAMMER

Break the message into digraphs:
[['H', 'A'], ['M', 'X'], ['M', 'E'], ['R', 'X']]

Matrix:
[['M', 'O', 'N', 'A', 'R'], ['C', 'H', 'Y', 'B', 'D'], ['E', 'F', 'G', 'I', 'K'], ['L', 'P', 'Q', 'S',

Cipher:
['B', 'O', 'A', 'U', 'C', 'L', 'A', 'Z']

fafasonga-2:Data_sec admin$ python playfair.py
Playfair Cipher
Choose :
1,Encrypting
2,Decrypting
2

Please input the key : MONARCHY
Please input the cipher text: BOAUCLAZ

Decrypting:
Cipher: BOAUCLAZ
Plaintext:
hammer
```