OneTimePad

March 25, 2022

[ ]:
```
from textwrap import wrap
import math, secrets, operator, random, hashlib, base64, numpy
from Crypto.Cipher import AES
from Crypto import Random

#TODO alphabet to be ASCII too.

def get_digit(number, n):
    return number // 10**n % 10

def str_to_lst(alphabet,s):
    return [alphabet.index(x) for x in s]

def lst_to_str(alphabet,lst):
    return ''.join([alphabet[x] for x in lst])

def num_to_bits(numstr,bit_length):  # input: a string that contains an decimal␣
 ↪integer & number of bits that we like to represent the decimal
    dec=int(numstr)
    result=""
    while dec != 0:
        remainder = dec % 2
        dec = dec // 2
        result = str(remainder) + result
    while len(result) < bit_length:     # add zeros until binary number is␣
 ↪represented by bit_length number of bits
        result = "0" + result
    return result

def bits_to_num(bits): #input: a string that contains a binary number
    length=len(bits)
    number=0
    for x in range(1,length+1):
        number+= int(bits[x-1])*2**(length-x)
    return number

def lst_to_bits(lst,length):
```

```
    return ''.join([num_to_bits(x,length) for x in lst])

def str_to_bits(alphabet,s,length):
    numLst = str_to_lst(alphabet,s)
    return lst_to_bits(numLst,length)

def bits_to_lst(bits,length):
    bitLst = wrap(bits,length)
    numLst=[]
    for number in bitLst:
        numLst.append(bits_to_num(number))
    return numLst

def bits_to_str(alphabet, bits , length):
    numLst = bits_to_lst(bits, length)
    return lst_to_str(alphabet, numLst)

def bits_to_bytes(s): #https://stackoverflow.com/a/32676625/17378708
     return int(s, 2).to_bytes((len(s) + 7) // 8, byteorder='big')

def bytes_to_bits(s): #https://stackoverflow.com/a/32675774/17378708
    return ''.join(format(byte, '08b') for byte in s)

def exclusiveOR(bits1,bits2):    # input: bits in string format
    result=""
    for i, j in zip(bits1, bits2):
        if i == j:
            result += "0"
        else:
            result += "1"
    return result
```

[ ]:
```
def encryptionOTP(plaintext):
    plain_bits= str_to_bits(alphabet,plaintext,5)
    key_bits= num_to_bits(format(secrets.
 ↪randbits(len(plain_bits))),len(plain_bits))
    key = bits_to_str(alphabet,key_bits,5)
    print("Random generated key:", key)
    cipher_bits = exclusiveOR(plain_bits,key_bits)
    ciphertext = bits_to_str(alphabet,cipher_bits,5)
    return ciphertext, key

def decryption(ciphertext,key):
    cipher_bits=str_to_bits(alphabet,ciphertext,5)
    key_bits=str_to_bits(alphabet,key,5)
    plain_bits= exclusiveOR(cipher_bits,key_bits)
    plaintext = bits_to_str(alphabet, plain_bits , 5)
```

```
    return plaintext
```