Vigenere tores

Egy elképzelt nyelvben csak az angol ábécé nagybetűi szerepelnek. A nyelv minden szövege úgy épül fel, hogy blokkokból áll: minden blokk egymástól függetlenül

80% valószínűséggel egy olyan 5 hosszú blokk, melyben csak magánhangzók szerepelnek (A, E, I, O, U)

15% valószínűséggel egy olyan 7 hosszú blokk, melyben csak mássalhangzók szerepelnek,

5% valószínűséggel a következő sztring: "KORE".

Valósítsunk meg egy olyan törő algoritmust, mely a fenti nyelvnek egy legalább 200 hosszú szövegének Vigenere-titkosítását vissza tudja fejteni (jó eséllyel). Használhatjuk a 2-3. órán tekintett módszert, ahol a titkos szöveg ismétléseit kellett detekálni. A Vigenere-kulcs egy 3 és 15 közti hosszúságú lista véletlen shiftekkel.

In [1]:
```
import random
import math
import copy
```

In [2]:
```
# karakterek kodolasa
def indexOfLetter(c):
    return ord(c) - 65
def letterWithIndex(i):
    return chr(i+65)
def shiftChar(c, shift):
    i = indexOfLetter(c)
    new = (i + shift) % 26
    return letterWithIndex(new)
```

In [3]:
```
# kulcs eloallitasa
def generateKey():
    L = []
    random_number = random.randint(3, 15)
    for i in range (0,random_number):
        shift_number = random.randint(0, 26)
        L.append(shift_number)
    return L
```

In [4]:
```
# Vigenere titkosítás -> orai kod
def vigenere(s, shiftList):
    ret = ""
    # Current position in the shiftList
    pos = 0
    n = len(shiftList)
    for c in s:
        new = shiftChar(c, shiftList[pos])
        ret += new
        # Add one, restart if list is over
        pos = (pos + 1) % n
    return ret
```

In [5]:
```
# Valoszinuseghez szamgenerator
def blockProbability():
    L = ""
    rand_prob = random.randint(1, 100)
    if ((rand_prob > -1) and (rand_prob < 81)):
        for i in range (0,5):
            rand_char = random.randint(0,4)
            m = ['A', 'E', 'I', 'O', 'U']
            L += m[rand_char]
    elif ((rand_prob > 80) and (rand_prob < 96)):
        for i in range (0,7):
            m = ['B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O
            rand_char = random.randint(0, len(m)-1)
            L += m[rand_char]
    else:
        L += "KORE"
    return L
```

In [6]:
```
# Titkos szoveg eloallitasa
def encodingText (s, L):
    for i in range (0, len(s)):
        s[i] = vigenere(s[i], L)
        print(s[i])
```

In [7]:
```
# Ismetlodesek kiszurese
def repetitionsInString(s, length = 3):
    n = len(s)
    ret = [(i,j) for i in range(0, n-length) for j in range(i+1, n-length+1
    return ret
```

In [8]:
```
s = []
for i in range (0,50):
    s.append(blockProbability())
L = generateKey()
```

In [9]:
```
# egy hosszu stringet keszitek a szoveg blokkokbol
complete = ""
for i in range (0, len(s)):
    complete += s[i]
print(complete)
```

```
IIUUUOOUEOUEIEUUIUUIKOREKOREAAIAUEUAOUQKTKWRZAAEUOKOREKOREUIIOOPTYKHQRIUIIU
UOUAUUAAOUEIIUEOIEUEAIIUOEIUAUEIUAIEUIIIUOEOAOAOAOKOREUOIIATBCXXOXAOUOAOIUE
IIEEIIOIOOUEUIEAKOREEEEOEEAEUENWSHJCOOAIAAUOOAAEAIEOIIAAOOIIIUKOREKOREPBKBZ
CFJOONNJQUIOAIEAOIEEEUIUIAIUA
```

In [10]:
```
# titkositas ket fele modon

# 1 - blokkonkent titkositom a szoveget -> minden blokk karaktereihez adot
r = copy.deepcopy(s)
encodingText(r, L)
```

```
AFZTM
GLZDG
MBNDM
MFZTA
CLWD
CLWD
SXNZM
WRFNM
IHYJOOE
SXJTG
CLWD
CLWD
MFNNG
HQDJZNW
ARNHM
MLZZM
MXFNM
WFNTW
GFJTW
SFNTG
WFZZM
WFZZA
WRNHA
MLJNS
GXTZG
CLWD
MLNHS
LYHWPLC
SLZNS
GFZDA
ABJHA
GFTNM
WRNDS
CLWD
WBJNW
WXJTW
FTXGBZT
GXNZS
MLTZS
WXNDG
AFFZG
GFNHM
CLWD
CLWD
HYPARZK
BLTMFGV
MFTZA
WXTHW
WBZHM
AXNTS
```

In [11]:
```
# 2 - az egesz szovegen egyben vegig megyek, es karakterenkent kodolok
complete_text = copy.deepcopy(complete)
vigenere(complete_text, L)
```

Out[11]:
```
'AFZTMLTTWLZDABZTARZHCLWDCLWDSXNZMBZZGRVJLHBQRXFDMLPNJBPNJBZHALTOLVPGIONTAF
ZTGRFTMXFNMBNHMBTHWRJZAFZNWFZZMBNTSFJTAFNTGBTZGXTZGHTQWRTHAXYAUUCNPXTTGXTHM
BNHWBNHGFTNMBZHWXPNJBJDWLJDSBZDFTXGBZTNSFFZMLTZSBFHWLNHSXTNAFNTCLWDCLWDHYPA
RZKIGLSMBNZHGXNDSLNDWBZHMFFHMX'
```

In [12]:
```
# ismerem a nyelvet, igy tudom, hogy vannak benne 4 hosszu KORE stringek, 
# a 4 hosszu stringeknel pontosan vissza tudom fejteni, hogy az elso 4 kar

four_char = repetitionsInString(complete_text, 7)
print(four_char)
```

```
[(20, 50), (20, 212), (21, 51), (21, 213), (50, 212), (51, 213)]
```

In [13]:
```
def guessedKeyLength(ciphertext, length = 3):
    differences = [j-i for (i,j) in repetitionsInString(ciphertext, length
    return gcd(differences)
```

In [14]:
```
# megprobalom a 4 hosszu ismetlodesekbol az eltolasokat meghatarozni
def KeyLengths(ciphertext, length = 3):
    differences = [j-i for (i,j) in repetitionsInString(ciphertext, length
    return differences
```

In [15]:
```
differences = KeyLengths(complete_text, 5)
print(differences)

def gcdOfKeys(array):
    b=array[0]
    for i in range (0, len(array)):
        s=math.gcd(b,array[i])
        b=s
    return b
print(gcdOfKeys(differences))
```

```
[30, 192, 30, 192, 30, 192, 30, 192, 75, 162, 162, 162, 162, 71, 17, 2]
1
```

In [30]:
```
m = ['A', 'E', 'I', 'O', 'U']
L2 = [3,2,1]
complete2 = copy.deepcopy(complete)
print(complete2)
complete2
complete2 = vigenere(complete2, L)
print(L)
print(complete2)
print(repetitionsInString(complete2, 4))
diff = KeyLengths(complete2, 4)
gcd_needed = gcdOfKeys(diff)

print(gcd_needed)
print(complete2[50:54])
print(complete2[125:129])
print(complete[50:54])
print(complete[125:129])
```

```
IIUUUOOUEOUEIEUUIUUIKOREKOREAAIAUEUAOUQKTKWRZAAEUOKOREKOREUIIOOPTYKHQRIUIIU
UOUAUUAAOUEIIUEOIEUEAIIUOEIUAUEIUAIEUIIIUOEOAOAOAOKOREUOIIATBCXXOXAOUOAOIUE
IIEEIIOIOOUEUIEAKOREEEEOEEAEUENWSHJCOOAIAAUOOAAEAIEOIIAAOOIIIUKOREKOREPBKBZ
CFJOONNJQUIOAIEAOIEEEUIUIAIUA
[18, 23, 5, 25]
AFZTMLTTWLZDABZTARZHCLWDCLWDSXNZMBZZGRVJLHBQRXFDMLPNJBPNJBZHALTOLVPGIONTAFZ
TGRFTMXFNMBNHMBTHWRJZAFZNWFZZMBNTSFJTAFNTGBTZGXTZGHTQWRTHAXYAUUCNPXTTGXTHMB
NHWBNHGFTNMBZHWXPNJBJDWLJDSBZDFTXGBZTNSFFZMLTZSBFHWLNHSXTNAFNTCLWDCLWDHYPAR
ZKIGLSMBNZHGXNDSLNDWBZHMFFHMX
[(0, 72), (20, 24), (20, 212), (20, 216), (21, 213), (22, 214), (23, 215),
(24, 212), (24, 216), (50, 54), (50, 166), (54, 166), (84, 148), (112, 208)
, (212, 216)]
4
PNJB
HTQW
KORE
KORE
```

In [17]:
```
# Kelloen kis kulcs eseten, mint a fenti L2 = 3 hosszu kulcs eseten a peri
# Tudjuk, hogy 5 hosszu esetben mindegyiknek 80%-os valoszinuseggel maganh
# Erdemes megvizsgalni, hogy melyik elem mennyire van maganhangzoktol, es
```

In [18]:
```
# statisztikai elofordulasok
LP = [0.8, 0.15, 0.05]
```