**Національний технічний університет України**

**“Київський політехнічний інститут”**

Лабораторна робота №2

**Криптоаналіз шифру Віженера**

**Виконали студенти:  
Групи ФІ-93  
Шашенок Микита  
Медведь Михайло**

**Варінт №11**

Київ 2022

**Мета роботи**

Набуття навичок програмної реалізації криптоаналізу шифру Віжінера, а саме: шифрування та дешифрування.

**Постановка задачі**

**Хід роботи**

1.Значення індексів відповідності, що були обраховані:

Для вихідного тексту :  


Для зашифрованих текстів із відповідними ключами:



2. Набори значень індексів відповідності,

одержаних при встановленні довжини ключа шифру Віженера:



3.Довжина та значення ключа, одержане шляхом співставлення найчастіших літер блоків найчастішій літері мови та одержане із використанням функції M\_i(g):



4. Фрагмент шифротексту та розшифрованого тексту відповідно за варіантом (5-10 рядочків) :

Шифротекст:



Розшифрований текст:



**Програмний код:**

from collections import Counter

alphabet = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'

all\_r = ['да', 'нет', 'мама', 'хорол', 'мамапапа', 'какутебядела']

m = 32

probability = [0.07657778115128833, 0.016759600394700733, 0.043686831825958, 0.0162574663249041,

0.029881648167502585, 0.08499728497276214, 0.011135698812978415, 0.014484816283156905,

0.06228797972779662, 0.009776433563575429, 0.03135418552102248, 0.044598847427146766,

0.030166580058270907, 0.061594334059286854, 0.11077194355078852, 0.024570704564165144,

0.03885233171210201, 0.05001839211999837, 0.06267800944712704, 0.028835340896484478,

0.001070830097682593, 0.008219817947205858, 0.002677659121031827, 0.017307276856874273,

0.008027138594842033, 0.002931061663231525, 0.0002382217447407295, 0.01616988480110236,

0.022518961399903076, 0.0029404036924370433, 0.005505958463002645, 0.020197467142331654]

def Read\_File(filename: str):

text = open(filename, 'r', encoding='utf-8')

texts = str(text.read())

return texts

def Encode(text, key):

real\_key, lenght, text = [], len(text), list(text)

for i in range (lenght):

real\_key.append(key[i % len(key)])

encrypt = [0]\*lenght

for j in range (lenght):

encrypt[j] = (ord(text[j]) + ord(real\_key[j]) - 2\*1072) % m

for k in range (lenght):

encrypt[k] = chr(encrypt[k] + 1072)

return "".join(encrypt)

def Decode(cryptotext, key):

n, k = len(cryptotext), len(key)

decrypt = [(ord(cryptotext[i]) - ord(key[i % k]) -2\*1072) % m for i in range(n)]

real\_message = [chr(decrypt[i] + 1072) for i in range(n)]

return "".join(real\_message)

def Match\_Index(text):

n, s, counter = len(text), 0, Counter(text)

for letter in alphabet:

s+= counter[letter]\*(counter[letter] - 1)

return s/(n\*(n - 1))

def Blocking(text, r):

return [text[i::r] for i in range(r)]

def Get\_Key\_Lenght(cryptotext):

max\_len, k = 30, 1

mean\_values =[]

while k < max\_len:

blocks = Blocking(cryptotext, k)

indexes = [Match\_Index(block) for block in blocks]

mean\_values.append(sum(indexes)/len(indexes))

k = k + 1

return mean\_values

def M(block, letter):

summ, counter = 0, Counter(block)

for i, ltr in enumerate(alphabet) :

summ += probability[i]\*counter[chr(((ord(ltr) + ord(letter) - 2\*1072) % m) + 1072)]

return summ

def Key\_Letter\_1(block):

results = [M(block, letter) for letter in alphabet]

return chr(results.index(max(results)) + 1072)

def Key\_Letter\_2(block):

max\_letter = Counter(block).most\_common(1)[0][0]

return chr(((ord(max\_letter) - ord('о') - 2\*1072) % m + 1072))

def Key\_1(crypto\_text,key\_lenght):

blocks = Blocking(crypto\_text, key\_lenght)

return ''.join([Key\_Letter\_1(block) for block in blocks])

def Key\_2(crypto\_text,key\_lenght):

blocks = Blocking(crypto\_text, key\_lenght)

return ''.join([Key\_Letter\_2(block) for block in blocks])

def main():

text = Read\_File('text\_for\_encoding.txt').replace('\n', '')

crypto\_text = Read\_File('variant.txt').replace('\n', '')

text\_for\_encoding = (''.join(filter(str.isalpha, text))).lower()

encoded\_texts = [(r,Encode(text\_for\_encoding, r)) for r in all\_r]

indexes\_r = [Match\_Index(cypher[1]) for cypher in encoded\_texts]

key\_lenght = Get\_Key\_Lenght(crypto\_text).index(max(Get\_Key\_Lenght(crypto\_text))) + 1

for j in range(len(Get\_Key\_Lenght(crypto\_text))):

print(f'Match index for blocks with period {j+1}:{Get\_Key\_Lenght(crypto\_text)[j]}')

key\_1, key\_2 = str(Key\_1(crypto\_text,key\_lenght )), str(Key\_2(crypto\_text,key\_lenght))

real\_message\_1, real\_message\_2 = Decode(crypto\_text, key\_1), Decode(crypto\_text, key\_2)

print(f'Match index for start text:{Match\_Index(text\_for\_encoding)}\n')

for i in range(len(all\_r)):

print(f'Match index for encoded text with key "{all\_r[i]}":, {indexes\_r[i]}')

print(f'This is the key lenght: {key\_lenght}\nThis is the key from comparison method:{key\_2}')

print(f'This is the key from M\_i(g) method :{key\_1}\n This is the text:{real\_message\_1}')

if \_\_name\_\_ == '\_\_main\_\_':

main()

Висновки

Наша бригада зробила програмну реалізацію криптоаналізу шифра Віжинера включно із взламом самого шифру. Було встановлено та експерементально перевірено, що текст, утворений шифром Віжинера зберігає статистичні властивості мови, якою він був написаний, завдяки чому і є достатньо простим його взлам за допомогою ідей так званого частотного аналізу, який базується на тому, що частоти символів тексту та шифротексту відміняються лише заміною самих літер.



**Код програми**