Ministerul Educaţiei și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

**RAPORT**

Lucrare de laborator Nr.3

*la Criptografie*

Tema: Cifruri polialfabetice

A efectuat: st. gr. SI-212 Șeremet Alexandru

A verificat: lect. asist. ictor Andronatiev

Chişinău 2022

**Sarcina:**

Sarcina 3.1. De implementat algoritmul Playfair în unul din limbajele de programare pentru mesaje în limba română (31 de litere). Valorile caracterelor textului sunt cuprinse între ‘A’ și ’Z’, ’a’ și ’z’ și nu sunt premise alte valori. În cazul în care utilizatorul introduce alte valori - i se va sugera diapazonul corect al caracterelor. Lungimea cheii nu trebuie să fie mai mică de 7. Utilizatorul va putea alege operația - criptare sau decriptare, va putea introduce cheia, mesajul sau criptograma și va obține criptograma sau mesajul decriptat. Faza finală de adăugare a spațiilor noi, în funcție de limba folosită și de logica mesajului – se va face manual.

Sarcină 3.2. De implementat algoritmul Vigenere în unul din limbajele de programare pentru mesaje în limba română (31 de litere), acestea fiind codificate cu numerele 0, 1, ... 30. Valorile caracterelor textului sunt cuprinse între ‘A’ și ’Z’, ’a’ și ’z’ și nu sunt premise alte valori. În cazul în care utilizatorul introduce alte valori - i se va sugera diapazonul corect al caracterelor. Lungimea cheii nu trebuie să fie mai mică de 7. Criptarea și decriptarea se va realiza în conformitate cu formulele din modelul matematic prezentat mai sus. În mesaj mai întâi trebuie eliminate spațiile, apoi toate literele se vor transforma în majuscule. Utilizatorul va putea alege operația - criptare sau decriptare, va putea introduce cheia, mesajul sau criptograma și va obține criptograma sau mesajul decriptat.

**Rezumat:**

3.1 Cifrul Vigenere

Cifrul Vigenere folosește aceleași operații ca și cifrul Cezar. Cifrul Vigenere și fel deplasează literele, dar, spre deosebire de Cezar, nu se poate sparge ușor în 26 combinații. Cifrul Vigenere folosește o deplasare multiplă. Cheia nu este constituită de o singură deplasare, ci de mai multe, fiind generate de câțiva întregi ki, unde 0 ≤ ki ≤ 25, dacă luăm ca reper alfabetul latin cu 26 de litere. Criptarea se face în felul următor:

ci = (mi + ki) mod 26.

Cheia poate fi, de exemplu, k = (5, 20, 17, 10, 20, 13) și ar provoca deplasarea primei litere cu 5, c1=m1 + 5 (mod 26), a celei de a doua cu 20, c2 = m2 + 20 (mod 26), ș.a.m.d. până la sfârșitul cheii și apoi de la început, din nou. Cheia este de obicei un cuvânt, pentru a fi mai ușor de memorat – cheia de mai sus corespunde cuvântului „furtun”. Metoda cu deplasare multiplă oferă protecție suplimentară din două motive:

• primul motiv este că ceilalți nu cunosc lungimea cheii;

• cel de al doilea motiv este că numărul de soluții posibile crește odată cu mărimea cheii; de exemplu, pentru lungimea cheii egală cu 5, numărul de combinații care ar fi necesare la căutarea exhaustivă ar fi 265 = 11 881 376.

3.2 Algoritmul de criptare Playfair

Din punct de vedere al criptografiei moderne, algoritmul de criptare Playfair este unul învechit, chiar primitiv. Orice calculator personal modern poate găsi (sparge) cheia și descifra mesajul într-un interval de timp de câteva secunde sau chiar sutimi de secunde, folosind softwareul potrivit. Unii dintre cei mai iscusiți criptanaliști sau chiar unii experți în cuvinte încrucișate pot sparge mesajul criptat în câteva minute folosind doar un creion și o foaie de hârtie.

Cu toate că este un algoritm depășit din toate punctele de vedere, algoritmul Playfair este unul dintre primii algoritmi ce folosește principiile moderne ale cifrurilor bloc. Studierea acestui algoritm vă poate oferi o mai bună înțelegere intuitivă a criptografiei moderne fără a folosi cunoștințe complexe de matematică sau teoria numerelor.

Descrierea generală a algoritmului Playfair

Criptarea Playfair implică parcurgerea următorilor pași:

a) pregătirea textului ce urmează a fi criptat;

b) construirea matricei de criptare;

c) construirea mesajului criptat

**Codul deplin al programului pentru cifrul Vigenere:**

def vigenere(

text: str,

key: str,

alphabet='AĂÂBCDEFGHIÎJKLMNOPQRSȘTȚUVWXYZ',

encrypt=True

):

try:

for i in text:

alphabet.index(i)

for i in key:

alphabet.index(i)

except ValueError:

print(

'textul si cheia trebuie sa fie compuse doar din litere ale alfabetului roman.')

return

if len(key) < 7:

print('cheia este prea mica, sunt necesare cel putin 7 caractere.')

return

result = ''

for i in range(len(text)):

letter\_n = alphabet.index(text[i])

key\_n = alphabet.index(key[i % len(key)])

if encrypt:

value = (letter\_n + key\_n) % len(alphabet)

else:

value = (letter\_n - key\_n) % len(alphabet)

result += alphabet[value]

return result

def vigenere\_encrypt(text, key):

return vigenere(text=text, key=key, encrypt=True)

def vigenere\_decrypt(text, key):

return vigenere(text=text, key=key, encrypt=False)

def main():

encrypt = True if input(

'doriti sa criptati sau decriptati? (1/0)') == '1' else False

key = input('cheie: ').replace(' ', '').upper()

text = input('text: ').replace(' ', '').upper()

print('rezultat:',

vigenere\_encrypt(text, key)

if encrypt else

vigenere\_decrypt(text, key))

main()

**Execuția codului:**



**Codul deplin al programului pentru algoritmul Playfair**

import numpy as np

import random

def playfair(

text: str,

key: str,

alphabet='AĂÂBCDEFGHIÎKLMNOPQRSȘTȚUVWXYZ',

obfuscation='QVXZ',

encrypt=True,

):

def checkValidity():

try:

for i in text:

alphabet.index(i)

for i in key:

alphabet.index(i)

except ValueError:

print(

'textul si cheia trebuie sa fie compuse doar din litere ale alfabetului roman.')

quit()

if len(key) < 7:

print('cheia este prea mica, sunt necesare cel putin 7 caractere.')

quit()

def processKey():

res = ''.join(dict.fromkeys(key + alphabet))

res = np.array(list(res))

res = np.reshape(res, (5, 6))

return res

checkValidity()

key = processKey()

if encrypt:

prev = ''

def obfuscatingChar():

nonlocal prev

res = ''

res = obfuscation[random.randint(0, len(obfuscation) - 1)]

while res == prev:

res = obfuscation[random.randint(0, len(obfuscation) - 1)]

prev = res

return res

def processText():

res = iter(text)

res = ' '.join(a+b for a, b in zip(res, res))

if len(text) % 2 != 0:

res += ' ' + text[len(text) - 1]

for i in range(len(res)):

if(res[i] == res[i - 1]):

res = res[:i] + obfuscatingChar() + res[i:]

res = res.replace(' ', '')

isLenOdd = False

if len(res) % 2 != 0:

isLenOdd = True

lastChar = res[len(res) - 1]

res = iter(res)

res = ' '.join(a+b for a, b in zip(res, res))

if isLenOdd:

res += ' ' + lastChar

isLenOdd = False

if len(res.replace(' ', '')) % 2 != 0:

res += obfuscatingChar()

return res.split()

text = processText()

expr = '+'

loopx = 0

loopy = 0

else:

def processText():

res = iter(text)

res = ' '.join(a+b for a, b in zip(res, res))

return res.split()

text = processText()

expr = '-'

loopx = len(key) - 1

loopy = len(key[0]) - 1

res = ''

for a, b in text:

a\_i = np.where(key == a)

b\_i = np.where(key == b)

if (b\_i[0] == a\_i[0]):

try:

res += str(key[(a\_i[0],

eval(str(int(a\_i[1])) + expr + '1'))])

except IndexError:

res += str(key[(a\_i[0], loopy)])

try:

res += str(key[(b\_i[0],

eval(str(int(b\_i[1])) + expr + '1'))])

except IndexError:

res += str(key[(b\_i[0], loopy)])

elif (b\_i[1] == a\_i[1]):

try:

res += str(key[(eval(str(int(a\_i[0])) + expr + '1'),

a\_i[1])])

except IndexError:

res += str(key[(loopx, a\_i[1])])

try:

res += str(key[(eval(str(int(b\_i[0])) + expr + '1'),

b\_i[1])])

except IndexError:

res += str(key[(loopx, b\_i[1])])

else:

res += str(key[(b\_i[0], a\_i[1])])

res += str(key[(a\_i[0], b\_i[1])])

res = res.replace('[', '').replace(']', '').replace('\'', '')

return res

def playfair\_encrypt(text, key):

return playfair(text=text, key=key, encrypt=True)

def playfair\_decrypt(text, key):

return playfair(text=text, key=key, encrypt=False)

def main():

encrypt = True if input(

'doriti sa criptati sau decriptati? (1/0) ') == '1' else False

key = input('cheie: ').upper().replace(' ', '').replace('J', 'I')

text = input('text: ').upper().replace(' ', '').replace('J', 'I')

print('rezultat:',

(playfair\_encrypt(text, key)

if encrypt else

playfair\_decrypt(text, key)))

main()

**Execuția codului:**



**Concluzii:**

* Cifrul Vigenere este o iterație mult mai sigură a algoritmului inventat de cezar. În loc să fie posibilă spargerea cu încercarea a doar 26 de chei pentru limba engleză, cheile posibile sunt o mulțime enormă, însă și cifrul dat este vulnerabil în multe feluri.
* Algoritmul Playfair prezintă un mod mai complicat de a cripta un mesaj, care, de asemenea, oferă și o securitate mai mare în comparație cu cifrul Cezar. Spre deosebire de cifrul Vigenere, care se bazează pe variația caracterului mesajului și caracterului cheii pentru a evita dublările literelor, cifrul Playfair necesită introducerea de către utilizator a caracterelor rar folosite între literele ce se dublează pentru a nu oferi un avantaj celui care dorește să-l spargă.