Ministerul Educaţiei și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

**RAPORT**

Lucrare de laborator Nr.1

*la Criptografie*

Tema: Cifruri de substituție monoalfabetică și permutare

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

A verificat: lect. asist. Victor Andronatiev

Chişinău 2022

**Sarcina:**

Sarcina 1.1. De implementat algoritmul Cezar pentru alfabetul limbii engleze în unul din limbajele de programare. Utilizați doar codificarea literelor cum este arătat în tabelul 1 (nu se permite de folosit codificările specificate în limbajul de programare, de ex. ASCII sau Unicode). Valorile cheii vor fi cuprinse între 1 și 25 inclusiv și nu se permit alte valori. 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. Înainte de criptare textul va fi transformat în majuscule și vor fi eliminate spațiile. Utilizatorul va putea alege operația - criptare sau decriptare, va putea introduce cheia, mesajul sau criptograma și va obține respectiv criptograma sau mesajul decriptat.

Sarcina 1.2. De implementat algoritmul Cezar cu 2 chei, cu păstrarea condițiilor exprimate în Sarcina 1.1. În plus, cheia 2 trebuie să conțină doar litere ale alfabetului latin, și să aibă o lungime nu mai mică de 7.

**Rezumat:**

1.1 Cifrul Cezar

Cifrul lui Cesar (sau Cezar). În acest cifru fiecare literă a textului clar este înlocuită cu o nouă literă obţinută printr-o deplasare alfabetică. Cheia secretă k, care este aceeaşi la criptare cât şi la decriptare, constă în numărul care indică deplasarea alfabetică, adică k{1, 2, 3,..., n–1}, unde n este lungimea alfabetului. Criptarea şi decriptarea mesajului cu cifrul Cezar poate fi definită de formulele

c = ek(x) = x + k (mod n),

m = dk(y) = y – k (mod n),

unde x și y sunt reprezentarea numerică a caracterului respectiv din textul clar m și din criptograma c.Funcţia numită Modulo (a mod b) returnează restul împărţirii numărului întreg a la numărul întreg b.Această metodă de criptare este numită aşa după Iulius Cezar, care o folosea pentru a comunicacu generalii săi, folosind cheia k = 3

1.2 Cifrul Cezar cu permutari

Având în vedere criptorezistența scăzută a cifrului Cezar, datorată în primul rând spațiului de chei, care constă doar din 25 de chei diferite pentru alfabetul latin, acesta poate fi spart prin încercarea consecutivă a tuturor cheilor. Dacă mesajul a fost criptat cu cifrul Cezar, atunci una dintre chei ne va da un text citibil în limba în care a fost scris mesajul.

Pentru a spori criptorezistența cifrului Cezar se poate de aplicat o permutare a alfabetului prin aplicarea unui cuvânt-cheie (a nu se confunda cu cheia de bază a cifrului). Această cheie poate fi orice consecutivitate de litere a alfabetului - fie un cuvânt din vocabular, fie unul fără sens.

**Codul deplin al programului:**

from asyncore import ExitNow

from audioop import add

import sys

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

latin\_alphabet = 'ABCDEFGHIKLMNOPQRSTVXYZ'

def isEnglish(s):

try:

s.encode(encoding='utf-8').decode('ascii')

except UnicodeDecodeError:

return False

else:

return True

def caesarLoop(string\_input, op, alphabet\_key):

done = False

while not done:

key = int(input("Introduceti cheia(1->25): "))

if (key > 0 and key < 26):

string\_res = ""

for i in range(len(string\_input)):

character = string\_input[i]

location\_of\_character = alphabet\_key.find(character)

new\_location = eval(

str(location\_of\_character) + op + str(key)) % 26

string\_res += alphabet\_key[new\_location]

done = True

else:

print(

"Valoare invalida, introduceti o cifra intre 1 si 25.")

print("\nRezultat: \n")

return string\_res

def getValidInput(message):

while True:

res = (

input(message)).upper().replace(' ', '')

if (isEnglish(res) and res.isalpha()):

break

else:

print(

"Valoare invalida, introduceti o propozitie ce contine doar litere din intervalele A-Z si a-z.")

return res

def caesar(op):

string\_input = getValidInput("introduceti o propozitie exclusiv alfabetica: ")

string\_res = caesarLoop(string\_input, op, alphabet)

print(string\_res)

def getKey():

done = False

while not done:

user\_key = getValidInput("Introduceti o cheie compusa din cel putin 7 litere ale alfabetului latin: ")

key = ""

for char in user\_key:

if char in latin\_alphabet and len(user\_key) >= 7:

done = True

else:

print(

"Cheia trebuie sa contina doar litere din alfabetul latin si cel putin 7, introduceti din nou.")

done = False

break

for char in user\_key:

if char in alphabet and char not in key:

key += char

for char in alphabet:

if char not in key:

key += char

return key

def caesarKey(op):

string\_input = getValidInput("introduceti o propozitie exclusiv alfabetica: ")

key = getKey()

print(caesarLoop(string\_input, op, key))

def quit():

print("iesire")

sys.exit()

def menu():

print("\n\*\*\*\*\*\*\*Meniul principal\*\*\*\*\*\*\*")

choice = input("""

1: Cezar encriptare

2: Cezar decriptare

3: Cezar + cuvant cheie

4: Cezar + cuvant cheie

0: Iesire din meniu

Alegeti o optiune

""")

match (choice):

case "1":

caesar('+')

case "2":

caesar('-')

case "3":

caesarKey('+')

case "4":

caesarKey('-')

case "0":

quit()

case \_:

print("alegerea nu este din meniu")

menu()

menu()

**Execuția codului:**











**Concluzii:**

* Cifrul cezar de sine stătător nu este tare eficient, deoarece poate fi spart ușor printr-un algoritm brute force, el având în total doar 25 de chei, pe de altă parte, un cifru cu cheie are o securitate sporită exponențial față de cifrul cezar simplu, iar dacă amândoua se unesc, securitatea criptării va crește și mai tare.
* În urma efectuării laboratorului, am înțeles bazele criptografiei și implementarea în cod a unui algoritm criptografic