**Ministerul Educației și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Lucrarea de laborator nr.4

*la Tehnologii ale Securității Informaționale*

Tema: Implementarea algoritmilor de criptare simetrică și asimetrică. Semnătura digitală.

A efectuat:

st. gr. TI-215 Vlașițchi Ștefan

A verificat:

asist. univ. Maxim Zalesciuc

Chișinău - 2023

**Scopul**

1. Algoritmul DES/AES

* Studierea algoritmului DES sau AES
* Modelul matematic și funcționalul
* Domeniile de aplicare

1. Algoritmul RSA

* Studierea modelului matematic al algoritmului RSA
* Generarea cheilor
* Domenii de aplicare
* Semnătura digitală

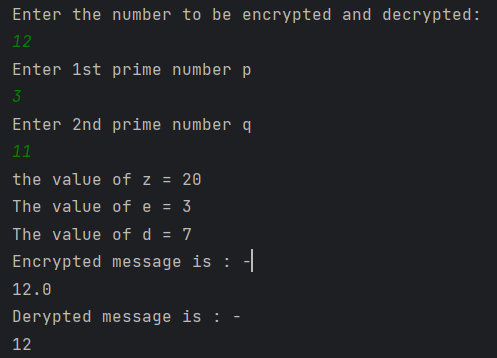
1. Realizarea aplicației

* Realizarea unei aplicații într-un limbaj de programare la alegere
* Interfața principală va reprezenta un meniu cu algoritmii DES/AES, RSA și semnătura digitală
* Opțiunile la alegere sunt: Implementarea DES sau AES, la semnătura digitală fie RSA sau DSA
* Limbaje de programare recomandate: C++, C#, Java, Python

**1. Algoritmul RSA**

import java.math.BigDecimal;  
import java.math.BigInteger;  
import java.util.Scanner;  
public class RSA  
{  
 public static void main(String args[])  
 {  
 Scanner sc = new Scanner(System.*in*);  
 int p,q,n,z,d=0,e,i;  
 System.*out*.println("Enter the number to be encrypted and decrypted: ");  
 int msg=sc.nextInt();  
 double c;  
 BigInteger msgback;  
 System.*out*.println("Enter 1st prime number p");  
 p=sc.nextInt();  
 System.*out*.println("Enter 2nd prime number q");  
 q=sc.nextInt();  
  
 n=p\*q;  
 z=(p-1)\*(q-1);  
 System.*out*.println("the value of z = "+ z);  
  
 for(e=2;e<z;e++)  
 {  
 if(*gcd*(e,z)==1)  
 {  
 break;  
 }  
 }  
 System.*out*.println("The value of e = "+ e);  
 for(i=0;i<=9;i++)  
 {  
 int x=1+(i\*z);  
 if(x%e==0)   
 {  
 d=x/e;  
 break;  
 }  
 }  
 System.*out*.println("The value of d = "+ d);  
 c=(Math.*pow*(msg,e))%n;  
 System.*out*.println("Encrypted message is : -");  
 System.*out*.println(c);  
 BigInteger N = BigInteger.*valueOf*(n);  
 BigInteger C = BigDecimal.*valueOf*(c).toBigInteger();  
 msgback = (C.pow(d)).mod(N);  
 System.*out*.println("Derypted message is : -");  
 System.*out*.println(msgback);  
 }  
 static int gcd(int e, int z)  
 {  
 if(e == 0)  
 return z;  
 else  
 return *gcd*(z % e, e);  
 }  
}

**Rezultatul:**



**2.Algoritmul DES/AES**

from Crypto.Cipher import AES, DES

from Crypto.PublicKey import RSA

from Crypto.Signature import DSS

from Crypto.Hash import SHA256

import PySimpleGUI as sg

# Creare fereastra principala

layout = [[sg.Text('Selecteaza un algoritm')],

[sg.Button('DES'), sg.Button('AES'), sg.Button('RSA'), sg.Button('DSA')]]

window = sg.Window('Criptare si semnatura digitala', layout)

# Functie pentru criptarea datelor utilizand algoritmul DES

def encrypt\_des(key, data):

cipher = DES.new(key, DES.MODE\_ECB)

return cipher.encrypt(data)

# Functie pentru decriptarea datelor utilizand algoritmul DES

def decrypt\_des(key, data):

cipher = DES.new(key, DES.MODE\_ECB)

return cipher.decrypt(data)

# Functie pentru criptarea datelor utilizand algoritmul AES

def encrypt\_aes(key, data):

cipher = AES.new(key, AES.MODE\_ECB)

return cipher.encrypt(data)

# Functie pentru decriptarea datelor utilizand algoritmul AES

def decrypt\_aes(key, data):

cipher = AES.new(key, AES.MODE\_ECB)

return cipher.decrypt(data)

# Functie pentru generarea cheii RSA si a perechii de chei publice si private

def generate\_rsa\_key():

key = RSA.generate(2048)

return key

# Functie pentru semnarea datelor utilizand algoritmul DSA

def sign\_dsa(key, data):

hash = SHA256.new(data)

signer = DSS.new(key, 'fips-186-3')

signature = signer.sign(hash)

return signature

# Functie pentru verificarea semnaturii digitale utilizand algoritmul DSA

**Concluzie**

Pentru realizarea unei astfel de aplicații, am putea utiliza limbajul de programare Python. Pentru implementarea algoritmilor de criptare DES și AES, putem folosi librăria criptografică PyCryptodome, care oferă o serie de funcții pentru criptarea și decriptarea datelor utilizând acești algoritmi.

Pentru semnătura digitală, putem alege între RSA sau DSA. Pentru implementarea algoritmului RSA, putem utiliza librăria criptografică PyCrypto, care oferă o serie de funcții pentru criptarea și decriptarea datelor utilizând acest algoritm. Pentru implementarea algoritmului DSA, putem utiliza librăria criptografică PyCryptodome, care oferă o serie de funcții pentru semnarea și verificarea semnăturilor digitale utilizând acest algoritm.