**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Lucrarea de laborator nr.4

*la Tehnologii a Securitatii Informationale*

A efectuat:

st. gr. TI-216 Rosca Dorin

A verificat: Zalesciuc Maxim

Chişinău – 2023

**Lucrare de laborator nr. 4**

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

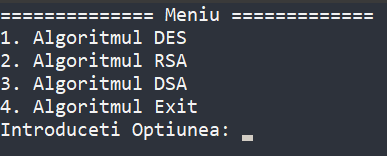
**Scopul lucrării:** Realizarea unei aplicații într-un limbaj de programare la alegere. Interfața principală va reprezenta un meniu cu algoritmii DES/AES, RSA și semnătura digital. Opțiunile la alegere sunt: Implementarea DES sau AES, la semnătura digitală fie RSA sau DSA

**1.Algoritmul DES (Data Encryption Standard) este un algoritm de criptare cu cheie simetrică, folosit pentru protejarea confidențialității datelor prin transformarea acestora într-o formă neinteligibilă pentru persoanele neautorizate. Algoritmul AES (Advanced Encryption Standard) este un alt algoritm de criptare cu cheie simetrică, considerat unul dintre cele mai sigure și rapide algoritme disponibile în prezent.**

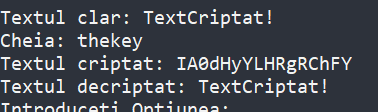
**2.** **Algoritmul RSA este un algoritm de criptare cu cheie publică, folosit pentru protejarea confidențialității datelor prin transformarea acestora într-o formă neinteligibilă pentru persoanele neautorizate. Algoritmul RSA se bazează pe utilizarea operațiilor matematice de factorizare a numerelor mari și este utilizat frecvent în cadrul comunicațiilor securizate online**

**3.Realizarea Aplicatiei:**

1. **Meniu:**

****

1. **Algoritmul DES**

****

**Cod:** static string Encrypt(string plainText, string key)

{

byte[] plainBytes = System.Text.Encoding.ASCII.GetBytes(plainText);

byte[] keyBytes = System.Text.Encoding.ASCII.GetBytes(key);

// Criptare

for (int i = 0; i < plainBytes.Length; i++)

{

plainBytes[i] = (byte)(plainBytes[i] ^ keyBytes[i % key.Length]);

}

return Convert.ToBase64String(plainBytes);

}

static string Decrypt(string encryptedText, string key)

{

byte[] encryptedBytes = Convert.FromBase64String(encryptedText);

byte[] keyBytes = System.Text.Encoding.ASCII.GetBytes(key);

// Decriptare

for (int i = 0; i < encryptedBytes.Length; i++)

{

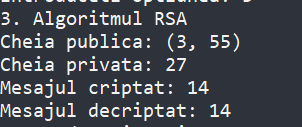
encryptedBytes[i] = (byte)(encryptedBytes[i] ^ keyBytes[i % key.Length]);

}

return System.Text.Encoding.ASCII.GetString(encryptedBytes);

}

1. **Algoritmul RSA**

****

**Cod:** static int ModInv(int a, int m)

{

int m0 = m, t, q;

int x0 = 0, x1 = 1;

if (m == 1)

return 0;

while (a > 1)

{

// q este catul impartirii a/m

q = a / m;

t = m;

// actualizarea m si a

m = a % m;

a = t;

// actualizarea x0 si x1

t = x0;

x0 = x1 - q \* x0;

x1 = t;

}

// asigurarea ca x1 este pozitiv

if (x1 < 0)

x1 += m0;

return x1;

}

static int ModPow(int a, int b, int m)

{

int res = 1;

while (b > 0)

{

// daca b este impar

if (b % 2 == 1)

res = (res \* a) % m;

// divizam b la 2

b /= 2;

a = (a \* a) % m;

}

return res;

}

1. **Algoritmul DSA**

****

public static void Method()

{

try

{

// Crearea unui obiect DSA

using (DSACryptoServiceProvider dsa = new DSACryptoServiceProvider())

{

// Generarea de chei

DSAParameters privateKey = dsa.ExportParameters(true);

DSAParameters publicKey = dsa.ExportParameters(false);

// Textul de semnat

string dataToSign = "Exemplu de text de semnat";

// Convertirea textului într-un array de bytes

byte[] dataToSignBytes = System.Text.Encoding.UTF8.GetBytes(dataToSign);

// Semnarea datelor

byte[] signature = SignData(dataToSignBytes, privateKey);

// Verificarea semnăturii

bool isVerified = VerifyData(dataToSignBytes, signature, publicKey);

if (isVerified)

{

Console.WriteLine("Semnătura este validă!");

}

else

{

Console.WriteLine("Semnătura nu este validă!");

}

}

}

catch (Exception e)

{

Console.WriteLine("A apărut o eroare: " + e.Message);

}

}

// Metoda pentru semnarea datelor

static byte[] SignData(byte[] data, DSAParameters privateKey)

{

using (DSACryptoServiceProvider dsa = new DSACryptoServiceProvider())

{

dsa.ImportParameters(privateKey);

return dsa.SignData(data);

}

}

// Metoda pentru verificarea semnăturii

static bool VerifyData(byte[] data, byte[] signature, DSAParameters publicKey)

{

using (DSACryptoServiceProvider dsa = new DSACryptoServiceProvider())

{

dsa.ImportParameters(publicKey);

return dsa.VerifyData(data, signature);

}

1. **Concluzie:**

Această lucrare de laborator a implicat implementarea algoritmilor de criptare simetrică și asimetrică, precum și a semnăturii digitale, cu ajutorul limbajului de programare ales. Algoritmii DES, AES, RSA și DSA au fost explicați în detaliu, iar aplicația realizată are o interfață intuitivă care permite utilizatorilor să selecteze algoritmii preferați și să cripteze/decripteze date sau să semneze digital documente.