Національний авіаційний університет

Факультет комп’ютерних наук та технологій

**ЗВІТ**

**по лабораторній роботі No 6**

**Асиметричні криптосистеми**

Дисципліна: «Криптологія»

Кафедра: прикладної математики

ОС: бакалавр

Спеціальність: 113 «Прикладна математика»

ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 3 курсу. 351 групи

Архіпов Олексій Тімурович

Перевірила: к.т.н.доц.Чолишкіна О.Г.

Київ 2023

**Тема**: Асиметричні криптосистеми. Алгоритм RSA.

**Мета**: Ознайомитись з математичними принципами функціонування алгоритму RSA. Навчитися проводити шифрування/дешифрування за допомогою даного алгоритму.

**Завдання**:

1. Розглянути загальні математичні принципи організації процедури

шифрування/дешифрування при використанні RSA.

2. Реалізувати програмно алгоритм шифрування і дешифрування методом RSA. Провести шифрування відкритого тексту.

**Короткі теоретичні відомості**

**Асиметричні криптосистеми** — ефективні системи криптографічного захисту даних, які також називають криптосистемами з відкритим ключем. В таких системах для зашифровування даних використовують один ключ, а для розшифровування — інший (звідси і назва — асиметричні). Перший ключ є *відкритим* і може бути опублікованим для використання усіма користувачами системи, які шифрують дані. Розшифровування даних за допомогою відкритого ключа неможливе. Для розшифровування даних отримувач зашифрованої інформації використовує другий ключ, який є секретним (*закритим*). Зрозуміло, що ключ розшифровування не може бути визначеним з ключа зашифровування.

Головне досягнення асиметричного шифрування в тому, що воно дозволяє людям, що не мають наперед наявної домовленості про безпеку, обмінюватися секретними повідомленнями. Необхідність відправникові й одержувачеві погоджувати таємний ключ по спеціальному захищеному каналу цілком відпала.

Алгоритм RSA (Rivest–Shamir–Adleman) - це асимметричний алгоритм шифрування, який використовує два ключа: відкритий і закритий. Він був розроблений в 1977 році Рональдом Рівестом, Аді Шаміром і Леонардом Адлеманом.

Алгоритм RSA містить наступні кроки:

Крок 1: Генерація ключів

1.1. Вибираються два простих числа p і q.

1.2. Обчислюється їх добуток n = p \* q.

1.3. Обчислюється значення функції Ейлера від числа n: φ(n) = (p - 1) \* (q - 1).

1.4. Вибирається ціле число e, таке, що 1 < e < φ(n), і e взаємно просто з φ(n). e стане відкритим ключем.

1.5. Обчислюється число d, зворотне по модулю φ(n) до числа e, тобто

d \* e = 1 (mod φ(n)). d стане закритим ключем.

1.6. Пара ключей (e, n) стає відкритим ключем, а пара ключей (d, n) стає закритим ключем.

Крок 2: Шифрування повідомлень

2.1. Перетворення тексту M в число m.

2.2. Шифрування числа m з використанням відкритого ключа (e, n): c = m^e mod n.

2.3. Зашифроване повідомлення c є шифротекстом.

Крок 3: Дешифрування повідомлень

3.1. Отриманий шифротекст c розшифровується за допомогою закритого ключа (d, n): m = c^d mod n.

3.2. Отримане число m відображається в повідомленні М.

Алгоритм RSA може використовуватися для шифрування та дешифрування повідомлень, а також для створення електронних підписів.



Рис.1. Алгоритм RSA

Алгоритм створення цифрового підпису за допомогою алгоритму RSA:

1. Обрати будь-яку хеш-функцію, наприклад SHA-256, для створення хеш-значення повідомлення, яке потрібно підписати.
2. Знайти хеш-значення повідомлення за допомогою обраної хеш-функції.
3. Зашифрувати хеш-значення повідомлення за допомогою зачиненого ключа d, використовуючи алгоритм RSA.
4. Отримане зашифроване значення і буде підписом для нашого повідомлення.
5. Для перевірки підпису, потрібно цей підпис розшифрувати за допомогою публічного ключа e, перевірити отриманий хеш з хешем повідомлення, якщо вони співпадають, то підпис е вірною.

**Робота програми**

Після запуску програми відкривається вікно, в якому потрібно або ввести параметри p,q, або натиснути кнопку Rand і ці числа згенеруються автоматично, також у текстбокс потрібно ввести повідомлення для створення підпису:

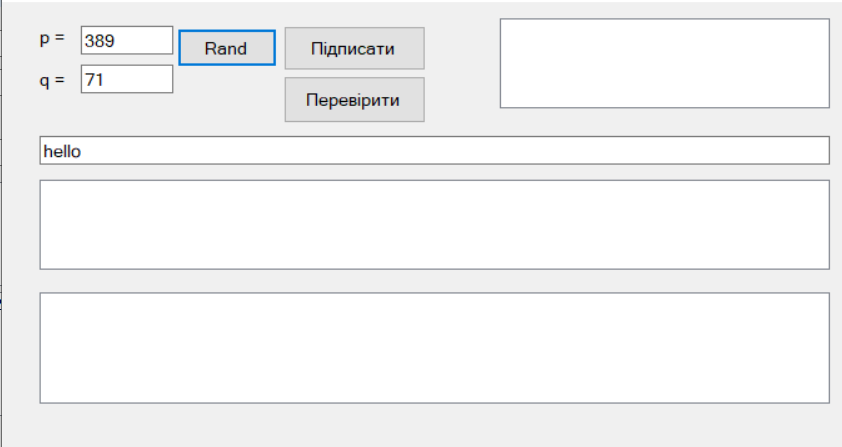


Рис.2. Результат згенерованих чисел

Після цього натискаємо Підписати:

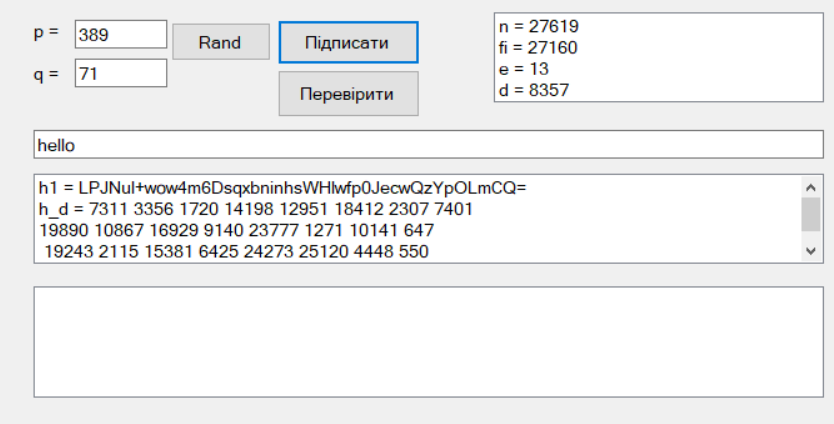


Рис.3. Підписане повідомлення

Програма поверне хеш повідомлення, зашифрований хеш, параметри n, fi, e, d.

Для перевірки підпису потрібно натиснути на кнопку Перевірити:

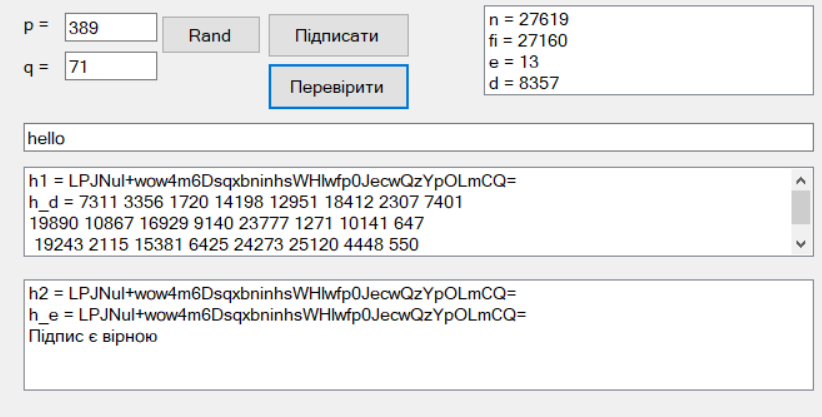


Рис.4. Перевірка підпису

І як ми бачимо, підпис є вірною, тож все було зроблено правильно.

Можу продемонструвати результат роботи програми коли ми при перевірці змінимо вхідне повідомлення, тоді підпис буде не вірним:

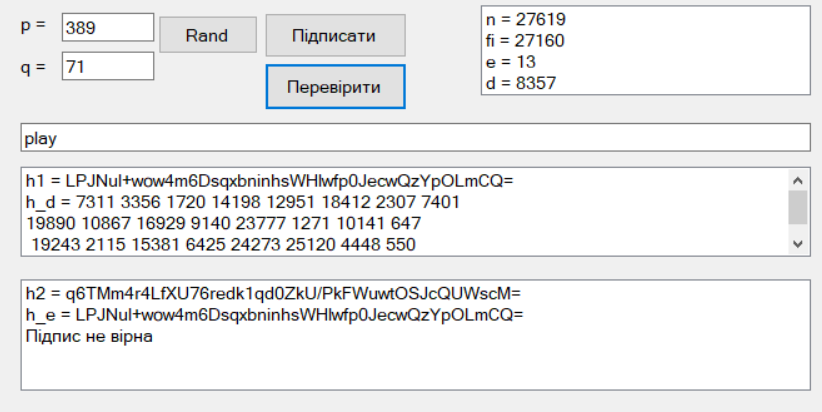


Рис.4. Неправильна підпис

**Висновки**

1. Ознайомлено з математичними принципами функціонування алгоритму RSA.
2. Вивчено як потрібно проводити шифрування/дешифрування за допомогою даного алгоритму.
3. Реалізував програмно алгоритм цифрового підпису методом RSA.

**Список використаної літератури**

1. Прикладна криптологія : системи шифрування : підручник / О. Г. Корченко, В. П. Сіденко, Ю. О. Дрейс. – К. : ДУТ, 2014. – 448 с.:іл.
2. Учасники проектів Вікімедіа. RSA – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/RSA.

**Додаток**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Security.Cryptography;

namespace lab6

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

List<int> m2;

int n, e, d;

public static bool IsPrimeNumber(uint n)

{

var result = true;

if (n > 1)

{

for (var i = 2u; i < n; i++)

{

if (n % i == 0)

{

result = false;

break;

}

}

}

else

{

result = false;

}

return result;

}

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

{

int res = 1;

a %= m;

while (b > 0)

{

if ((b & 1) != 0)

res = (res \* a) % m;

a = (a \* a) % m;

b >>= 1;

}

if (res < 0)

res += m;

return res;

}

void Generate()

{

int n = 500;

List<int> x = new List<int>();

for (var i = 0u; i < n; i++)

{

if (IsPrimeNumber(i))

{

x.Add(Convert.ToInt32(i));

}

}

Random r = new Random();

int j,k = 1;

do

{

j = r.Next(x.Count);

//k = r.Next(x.Count);

} while (j < 3);

if (x[j] > 100)

{

do

{

k = r.Next(x.Count);

} while (x[k] > 100);

}

else

{

do

{

k = r.Next(x.Count);

} while (x[k] < 100);

}

tbp.Text = x[j].ToString();

tbq.Text = x[k].ToString();

}

public static bool AreRelativelyPrime(int a, int b)

{

int gcd = 1;

for (int i = 1; i <= a && i <= b; ++i)

{

if (a % i == 0 && b % i == 0)

gcd = i;

}

return gcd == 1;

}

public static int FindD(int p, int q, int e)

{

int phi = (p - 1) \* (q - 1);

int d = 0;

int k = 1;

while (true)

{

d = (k \* phi + 1) / e;

if (d \* e == k \* phi + 1)

return d;

k++;

}

}

void Calc()

{

listBox1.Items.Clear();

listBox3.Items.Clear();

Random r = new Random();

string text = tbtext.Text;

byte[] hashBytes = new SHA256Managed().ComputeHash(Encoding.UTF8.GetBytes(text));

string s = Convert.ToBase64String(hashBytes);

listBox1.Items.Add("h1 = " + s);

int p = Convert.ToInt32(tbp.Text);

int q = Convert.ToInt32(tbq.Text);

n = p \* q;

int f = (p - 1) \* (q - 1);

List<int> e1 = new List<int>();

List<int> x = new List<int>();

bool t;

for (var i = 0u; i < f; i++)

{

if (IsPrimeNumber(i))

{

x.Add(Convert.ToInt32(i));

t = AreRelativelyPrime(x[x.Count - 1], f);

if (t == true) e1.Add(x[x.Count - 1]);

if (e1.Count == 10) break;

}

}

int e2 = r.Next(0, e1.Count - 1);

e = e1[e2];

d = FindD(p, q, e);

m2 = Encrypt\_RSA(hashBytes, d, n);

listBox3.Items.Add("n = " + n.ToString());

listBox3.Items.Add("fi = " + f.ToString());

listBox3.Items.Add("e = " + e.ToString());

listBox3.Items.Add("d = " + d.ToString());

}

List<int> Encrypt\_RSA(byte[] t1, int d1, int n1)

{

byte[] text = t1;

List<int> c = new List<int>();

string c1 = "";

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

{

c.Add(ModPow(text[i], d1, n1));

if (i != text.Length - 1) c1 += c[i].ToString() + " ";

else c1 += c[i].ToString();

}

int l = (int)(c1.Length / 4.0);

string firstHalf = c1.Substring(0, l);

string secondHalf = c1.Substring(l, l);

string thirdHalf = c1.Substring(2 \* l, l);

string forthHalf = c1.Substring(3 \* l);

listBox1.Items.Add("h\_d = " + firstHalf);

listBox1.Items.Add(secondHalf);

listBox1.Items.Add(thirdHalf);

listBox1.Items.Add(forthHalf);

return c;

}

string Decrypt\_RSA(List<int> t1, int e1, int n1)

{

byte[] m2 = new byte[t1.Count];

for (int i = 0; i < t1.Count; i++)

{

m2[i] = Convert.ToByte(ModPow(t1[i], e1, n1));

}

string s = Convert.ToBase64String(m2);

listBox2.Items.Add("h\_e = " + s);

return s;

}

void Calc2()

{

listBox2.Items.Clear();

string text = tbtext.Text;

byte[] hashBytes = new SHA256Managed().ComputeHash(Encoding.UTF8.GetBytes(text));

string s = Convert.ToBase64String(hashBytes);

listBox2.Items.Add("h2 = " + s);

string s1 = Decrypt\_RSA(m2, e, n);

if(s1 == s)

{

listBox2.Items.Add("Підпис є вірною");

}

else

{

listBox2.Items.Add("Підпис не вірна");

}

}

private void btnRand\_Click(object sender, EventArgs e)

{

Generate();

}

private void btnCalc\_Click(object sender, EventArgs e)

{

Calc();

}

private void btnDecrypt\_Click(object sender, EventArgs e)

{

Calc2();

}

}

}