**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра комп’ютеризованих систем захисту інформації

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №1**

**«АЛГОРИТМ RSA»**

**КИЇВ – 2022**

**Тема:** АЛГОРИТМ RSA

**Мета:** отримати навички у створенні програмної реалізації алгоритму асиметричного алгоритму RSA.

**Основні завдання:**

1. Розробити об’єктно-орієнтованою мовою програмування консольний або віконний додаток, що реалізує шифрування та дешифрування вмісту текстового або виконуваного файлу з використанням асиметричного алгоритму RSA. Програма повинна запитувати ім’я вхідного і вихідного файлів, тип шифру та вхідні параметри для проведення криптографічних перетворень. В якості вхідних параметрів виступають: e – відкритий ключ шифрування та n – добуток двох взаємнопростих чисел p та q . З використанням розширеного алгоритму Еквкліда розрахувати значення d – закритий ключ дешифрування.

2. Представити блок структурну схему своєї програмної реалізації.

3. Зробити висновки на підставі проведених теоретичних та практичних досліджень. У висновках слід зазначити, які навички та знання отримано під час виконання завдань.

**Основні теоретичні відомості**

RSA **-** перший повноцінний алгоритм з відкритим ключем, який можна використовувати і для шифрування, і для створення цифрових підписів, алгоритм RSA. Стійкість RSA базується на складності факторизації великих цілих чисел. Відкритий і закритий ключі є функціями двох великих простих чисел розрядністю 100…200 десяткових цифр і навіть більше. Відновлення відкритого тексту за шифртекстом та відкритим ключем є рівнозначне до розкладання числа на два великі прості множники. 1993 року алгоритм RSA було ухвалено за стандарт PKCS # 1: RSA Encryption Standard.

Для шифрування використовується проста операція піднесення до степеня за модулем . Для дешифрування ж необхідно обчислити функцію Ейлера від числа n, для цього необхідно знати розкладання числа  на прості множники (в цьому і полягає завдання факторизації).

Розглянемо основні етапи алгоритму RSA:

1. У довільний спосіб обираються два великі взаємно прості числа та . Обчислюється добуток *.*
2. Обчислюється функція Ейлера: *.*
3. Довільно обирається просте число – ключ зашифровування, яке задовольняє умовам ; .
4. Обчислюється число *–* ключ розшифровування, яке є оберненим до числа , тобто .
5. Пара чисел  виступає в якості відкритого ключа і передається по відкритих незахищених каналах зв’язку, а числа  – знищують для того щоб унеможливити компрометацію закритого ключа,  – закритий ключ дешифрування, який зберігається в секреті.

При шифруванні повідомлення спочатку розкладаємо на цифрові блоки, чиї розміри є менше за , тобто якщо та є 100-розрядними простими числами, то міститиме близько 200 розрядів і кожен блок повідомлення повинен мати близько 200 розрядів у довжину. Зашифроване повідомлення складатиметься з блоків такої самої довжини. Формула шифрування є досить простою:

.

Дешифрування забезпечується операцією піднесення до степеня за модулем одержаного шифротексту :

.

Система RSA використовується для захисту програмного забезпечення та в схемах цифрового підпису. Крім того, алгоритм використовується у великій кількості криптографічних додатків і прикладних протоколів: SSL; IPSec; IKE; SILC; SSH.

В апаратному виконанні RSA алгоритм застосовується в захищених телефонах, на мережевих платах Ethernet, на смарт-картах, широко використовується в криптографическом обладнанні Zaxus Datacryptor.

Також вона використовується у відкритій системі шифрування PGP та інших системах шифрування в поєднанні з симетричними алгоритмами.

Через низьку швидкість шифрування(близько30 кбіт/с при 512 бітному ключі на процесорі 2 ГГц), повідомлення зазвичай шифрують за допомогою більш продуктивних симетричних алгоритмів з випадковим сеансовим ключем (наприклад, AES, IDEA, Serpent, Twofish), а за допомогою RSA шифрують лише цей ключ, таким чином реалізується гібридна криптосистема. Такий механізм має потенційні уразливості зважаючи на необхідність використовувати криптографічно стійкий генератор псевдовипадкових чисел для формування випадкового сеансового ключа симетричного шифрування.

**Хід виконання та результати роботи.**

Початок програми нічим не особливий, це просто запуск форми в якій вже будуть проводиться обрахунки.

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Threading.Tasks;*

*using System.Windows.Forms;*

*namespace protect\_inf\_LR1*

*{*

*static class Program*

*{*

*static void Main()*

*{*

*Application.EnableVisualStyles();*

*Application.SetCompatibleTextRenderingDefault(false);*

*Application.Run(new Form1());*

*}*

*}*

**Програма починається з стартового екрану в якому треба ввести прості числа. Як початкові числа було обрано 7 та 11. M = 2, e = 23.**

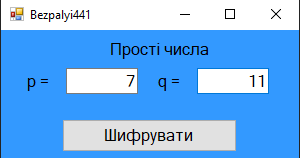


Рисунок 1 – початкові числа

**Як початкове повідомлення яке буде шифруватися 2.**



Рисунок 2 – початковий текст

*public partial class Form1 : Form*

*{*

*char[] characters = new char[] { '#', 'А', 'Б', 'В', 'Г',* *'ґ' 'Д', 'Е', 'Є', 'Ж', 'З', 'И', 'І', 'Ї','Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С','Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь','Ю', 'Я', ' ', '1', '2', '3', '4', '5', '6', '7','8', '9', '0' };*

**Далі це повідомлення потрапляє в клас шифрування.**

*private void buttonEncrypt\_Click(object sender, EventArgs e)*

*{*

*if ((textBox\_p.Text.Length > 0) && (textBox\_q.Text.Length > 0))*

*{*

*long p = Convert.ToInt64(textBox\_p.Text);*

*long q = Convert.ToInt64(textBox\_q.Text);*

*if (IsTheNumberSimple(p) && IsTheNumberSimple(q))*

*{*

*string s = "";*

*StreamReader sr = new StreamReader("in.txt");*

*while (!sr.EndOfStream)*

*{*

*s += sr.ReadLine();*

*}*

*sr.Close();*

*s = s.ToUpper();*

*long n = p \* q;*

*long m = (p - 1) \* (q - 1);*

*long d = Calculate\_d(m);*

*long e\_ = Calculate\_e(d, m);*

*List<string> result = RSA\_Endoce(s, e\_, n);*

*StreamWriter sw = new StreamWriter("out1.txt");*

*foreach (string item in result)*

*sw.WriteLine(item);*

*sw.Close();*

*textBox\_d.Text = d.ToString();*

*textBox\_n.Text = n.ToString();*

*Process.Start("out1.txt");*

*}*

*else*

*MessageBox.Show("p или q - не прості числа!");*

*}*

*else*

*MessageBox.Show("Введіть p и q!");*

*}*

**

Рисунок 3 – шифроване повідомлення

Далі виводиться зашифроване повідомлення.

**Після зашифроване повідомлення попадає в клас з дешифруванням.**

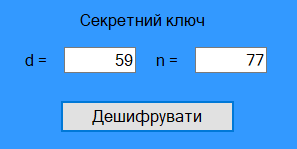
****

Рисунок 4 – введення секретних ключів

*private void buttonDecipher\_Click(object sender, EventArgs e)*

*{*

*if ((textBox\_d.Text.Length > 0) && (textBox\_n.Text.Length > 0))*

*{*

*long d = Convert.ToInt64(textBox\_d.Text);*

*long n = Convert.ToInt64(textBox\_n.Text);*

*List<string> input = new List<string>();*

*StreamReader sr = new StreamReader("out1.txt");*

*while (!sr.EndOfStream)*

*{*

*input.Add(sr.ReadLine());*

*}*

*sr.Close();*

*string result = RSA\_Dedoce(input, d, n);*

*StreamWriter sw = new StreamWriter("out2.txt");*

*sw.WriteLine(result);*

*sw.Close();*

*Process.Start("out2.txt");*

*}*

*else*

*MessageBox.Show("Введіть секретний ключ!");*

*}*

**Після чого воно дешифрується і на виході отримуємо знову початкові дані.**



Рисунок 5 – дешифроване повідомлення

**Блок-схема**



Рисунок 6 – блок-схема

**Загальний код**

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Threading.Tasks;*

*using System.Windows.Forms;*

*namespace protect\_inf\_LR1*

*{*

*static class Program*

*{*

*/// <summary>*

*/// Главная точка входа для приложения.*

*/// </summary>*

*[STAThread]*

*static void Main()*

*{*

*Application.EnableVisualStyles();*

*Application.SetCompatibleTextRenderingDefault(false);*

*Application.Run(new Form1());*

*}*

*}*

*}*

*public partial class Form1 : Form*

*{*

*char[] characters = new char[] { '#', 'А', 'Б', 'В', 'Г',* *'ґ' 'Д', 'Е', 'Є', 'Ж', 'З', 'И', 'І', 'Ї','Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С','Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь','Ю', 'Я', ' ', '1', '2', '3', '4', '5', '6', '7','8', '9', '0' };*

*public Form1()*

*{*

*InitializeComponent();*

*}*

*//зашифровати*

*private void buttonEncrypt\_Click(object sender, EventArgs e)*

*{*

*if ((textBox\_p.Text.Length > 0) && (textBox\_q.Text.Length > 0))*

*{*

*long p = Convert.ToInt64(textBox\_p.Text);*

*long q = Convert.ToInt64(textBox\_q.Text);*

*if (IsTheNumberSimple(p) && IsTheNumberSimple(q))*

*{*

*string s = "";*

*StreamReader sr = new StreamReader("in.txt");*

*while (!sr.EndOfStream)*

*{*

*s += sr.ReadLine();*

*}*

*sr.Close();*

*s = s.ToUpper();*

*long n = p \* q;*

*long m = (p - 1) \* (q - 1);*

*long d = Calculate\_d(m);*

*long e\_ = Calculate\_e(d, m);*

*List<string> result = RSA\_Endoce(s, e\_, n);*

*StreamWriter sw = new StreamWriter("out1.txt");*

*foreach (string item in result)*

*sw.WriteLine(item);*

*sw.Close();*

*textBox\_d.Text = d.ToString();*

*textBox\_n.Text = n.ToString();*

*Process.Start("out1.txt");*

*}*

*else*

*MessageBox.Show("p или q - не прості числа!");*

*}*

*else*

*MessageBox.Show("Введіть p и q!");*

*}*

*//дешифрувати*

*private void buttonDecipher\_Click(object sender, EventArgs e)*

*{*

*if ((textBox\_d.Text.Length > 0) && (textBox\_n.Text.Length > 0))*

*{*

*long d = Convert.ToInt64(textBox\_d.Text);*

*long n = Convert.ToInt64(textBox\_n.Text);*

*List<string> input = new List<string>();*

*StreamReader sr = new StreamReader("out1.txt");*

*while (!sr.EndOfStream)*

*{*

*input.Add(sr.ReadLine());*

*}*

*sr.Close();*

*string result = RSA\_Dedoce(input, d, n);*

*StreamWriter sw = new StreamWriter("out2.txt");*

*sw.WriteLine(result);*

*sw.Close();*

*Process.Start("out2.txt");*

*}*

*else*

*MessageBox.Show("Введіть секретний ключ!");*

*}*

*//перевірка: простое число?*

*private bool IsTheNumberSimple(long n)*

*{*

*if (n < 2)*

*return false;*

*if (n == 2)*

*return true;*

*for (long i = 2; i < n; i++)*

*if (n % i == 0)*

*return false;*

*return true;*

*}*

*//зашифровати*

*private List<string> RSA\_Endoce(string s, long e, long n)*

*{*

*List<string> result = new List<string>();*

*BigInteger bi;*

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

*{*

*int index = Array.IndexOf(characters, s[i]);*

*bi = new BigInteger(index);*

*bi = BigInteger.Pow(bi, (int)e);*

*BigInteger n\_ = new BigInteger((int)n);*

*bi = bi % n\_;*

*result.Add(bi.ToString());*

*}*

*return result;*

*}*

*//дешифрувати*

*private string RSA\_Dedoce(List<string> input, long d, long n)*

*{*

*string result = "";*

*BigInteger bi;*

*foreach (string item in input)*

*{*

*bi = new BigInteger(Convert.ToDouble(item));*

*bi = BigInteger.Pow(bi, (int)d);*

*BigInteger n\_ = new BigInteger((int)n);*

*bi = bi % n\_;*

*int index = Convert.ToInt32(bi.ToString());*

*result += characters[index].ToString();*

*}*

*return result;*

*}*

*//обрахунок параметра d. d повинно бути взаємно простим з m*

*private long Calculate\_d(long m)*

*{*

*long d = m - 1;*

*for (long i = 2; i <= m; i++)*

*if ((m % i == 0) && (d % i == 0)) //якщо мають спільні дільники*

*{*

*d--;*

*i = 1;*

*}*

*return d;*

*}*

*//обчислення параметра e*

*private long Calculate\_e(long d, long m)*

*{*

*long e = 25;*

*while (true)*

*{*

*if ((e \* d) % m == 1)*

*break;*

*else*

*e++;*

*}*

*return e;*

*}*

*private void Form1\_Load(object sender, EventArgs e)*

*{*

*}*

*private void label3\_Click(object sender, EventArgs e)*

*{*

*}*

*}*

}

**Висновок:** В результаті роботи було отримано програму в якій реалізовано програмне шифрування за методом RSA. Залежно від структури ключів, що використовуються, методи шифрування поділяються на:

симетричне шифрування: стороннім особам може бути відомий алгоритм шифрування, але невідома невелика порція секретної інформації - ключа, однакового для відправника та одержувача повідомлення; Приклади: DES, 3DES, AES, Blowfish, Twofish, ГОСТ 28147-89

асиметричне шифрування: стороннім особам може бути відомий алгоритм шифрування, і, можливо, відкритий ключ, але невідомий закритий ключ, відомий тільки одержувачу. Криптографічні системи з відкритим ключем в даний час широко застосовуються в різних мережевих протоколах, зокрема, в протоколах TLS і його попереднику SSL (що лежать в основі HTTPS), а також SSH, PGP, S/MIME і т.д. використовує асиметричне шифрування – ГОСТ Р 34.10-2001.

На даний момент асиметричне шифрування на основі відкритого ключа RSA (розшифровується, як Rivest, Shamir and Aldeman – творці алгоритму) використовує більшість продуктів на ринку інформаційної безпеки.

Його криптостійкость грунтується на складності розкладання на множники великих чисел, саме - виняткової труднощі завдання визначити секретний ключ виходячи з відкритого, оскільки цього потрібно вирішити завдання існування дільників цілого числа. Найбільш криптостійкі системи використовують 1024-бітові та великі числа.