**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра комп’ютеризованих систем захисту інформації

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №2**

**«АЛГОРИТМ ЕЛЬ–ГАМАЛЯ»**

**КИЇВ – 2022**

**Тема:** АЛГОРИТМ ЕЛЬ–ГАМАЛЯ

**Мета:** отримати навички у створенні програмної реалізації алгоритму асиметричного алгоритму Ель-Гамаля.

**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 Ель-Гамаля. Програма повинна запитувати ім'я вхідного і вихідного файлів, тип шифру та вхідні параметри для проведення криптографічних перетворень. В якості вхідних параметрів виступають:  – велике просте число,  – первісний корінь простого числа ;  – значення закритого ключа.
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**

Алгоритму Ель-Гамаля альтернатива алгоритму RSA і при рівному значенні ключа забезпечує таку саму криптостійкість. Може використовуватися для формування електронного підпису або для шифрування даних. Безпека алгоритму Ель–Гамаля базується на складності обчислювання дискретних логарифмів.

Розглянемо основні етапи алгоритму Ель–Гамаля:

Учасники інформаційного процесу обирають просте число і ціле число , який є первісним коренем за модулем , , .

Сторона *А* генерує сеансовий ключ  при умову, що .

Сторона *В* обирає число при умову, що  та розраховує парамеметр . Таким чином комбінація представляє собою відкритий ключ отримувача.

При виборі *х* і *у* одержувачем і відправником відповідно, природно, повинно виконуватися вимога до їх інформаційної ємності. Для генерації цих чисел повинен використовуватися *криптостійкий генератор псевдовипадкових чисел (КГПВЧ)*. В іншому випадку зловмисник просто визначить *х* або *у* повним перебором.

Далі за допомогою відкритого ключа відбувається шифрування повідомлення у за формулами.

; 

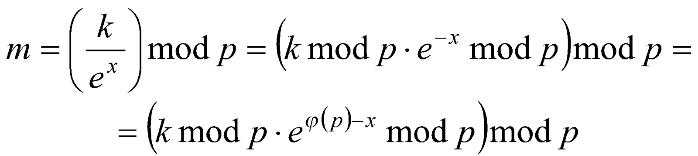
Отже, пара чисел і виступає в якості шифротексту.

Для проведення процедури дешифрування Одержувач, використовуючи свій закритий ключ дешифрує повідомлення вирішуючи Діофантове рівнянь:



 Для практичних обчислень більше підходить наступна формула:

 або



Алгоритм Ель–Гамаля – перший криптографічний алгоритм з відкритим ключем, який використовується для шифрування повідомлень і цифрових підписів, використання якого не обмежено патентами США. На відміну від RSA алгоритм Ель-Гамаля не запатентований і, тому, став більш дешевою альтернативою, оскільки не була потрібна оплата внесків за ліцензію. Вважається, що алгоритм потрапляє під дію патенту Діффі-Хеллмана. Існує велика кількість алгоритмів, заснованих на схемі Ель-Гамаля: це алгоритми DSA, ECDSA, KCDSA, схема Шнорра.

По криптостійкості в схемі Ель-Гамаль 512-бітове число *р* прирівнюється до 56-бітного симетричного ключа. Тому на практиці застосовуються *р* довжиною в 768, 1024,1536, 2048 біт.

**Хід виконання та результати роботи.**

**Блок-схема**



Рисунок 1 – блок-схема

На початку програми обирається випадковий примітивний елемент g поля Zp.

*public static bool Search\_g(int p, int g)*

*{*

*bool boolean = false;*

*List<BigInteger> array\_mod\_number = new List<BigInteger>();*

*BigInteger integer = ((BigInteger.Pow(g, 1)) % p);*

*array\_mod\_number.Add(integer);*

*for (int i = 2; i != p; i++)*

*{*

*integer = BigInteger.Pow(g, i) % p;*

*for (int j = 0; j != i - 1; j++)*

*{*

*if (array\_mod\_number[j] == integer)*

*{*

*g--;*

*array\_mod\_number.Clear();*

*i = 1;*

*integer = BigInteger.Pow(g, 1) % p;*

*array\_mod\_number.Add(integer);*

*break;*

*}*

*if ((j == i - 2) && (array\_mod\_number[j] != integer))*

*{*

*array\_mod\_number.Add(integer);*

*}*

*}*

*}*

*g\_main = g;*

*boolean = true;*

*return boolean;*

*}*

Обирається число p. Оскільки число p нам дано то воно становить **601.**

*public static int Search\_p()*

*{*

*Random random = new Random();*

*int p = 0;*

*Boolean boolean = false;*

*do*

*{*

***p = random.Next(601, 602);***

*for (int i = 2; i != p; i++)*

*{*

*if (i == p - 1)*

*{*

*boolean = Search\_g(p, p - 1);*

*break;*

*}*

*if (p % i == 0)*

*{*

*break;*

*}*

*}*

*}*

*while (boolean == false);*

*return p; }*

Далі іде сам метод шифрування.

*public static List<BigInteger> Cipher(string text, int p, BigInteger y)*

*{*

*List<BigInteger> array = new List<BigInteger>();*

*Random random = new Random();*

*int k = random.Next(1, p - 1);*

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

*{*

*a = BigInteger.Pow(g\_main, k) % p;*

*array.Add((BigInteger.Pow(y, k) \* (int)text[i]) % p);*

*}*

*return array;*

*}*

Після нього іде метод дешифрування.

*public static string Cipher\_RAZ(int length\_text, List<BigInteger> array\_number, int x, int p)*

*{*

*string save\_text = "";*

*BigInteger integer;*

*for (int i = 0; i != length\_text; i++)*

*{*

*integer = (array\_number[i] \* (BigInteger.Pow(a, p - 1 - x))) % p;*

*save\_text += (char)integer;*

*}*

*return save\_text;*

*}*

Вже далі після запуску програми вода відображає всі дані

*static void Main(string[] args)*

*{*

*int p = 0;*

*Random random = new Random();*

*p = Search\_p();*

*int x = 5; //закритий ключ*

*BigInteger y = BigInteger.Pow(g\_main, x) % p; //Відкритий ключ*

*Console.Write($"Числа [{p}, {g\_main}, {y}] ---> [p, g, y]\n");*

*Console.Write($"Відкритий ключ - это <y>\n");*

*Console.Write($"Закритий ключ: {x}\n");*



Рисунок 2 – всі початкові дані

Після цього програма зчитує початкове повідомлення. За завданням початковим повідомленням виступає дата народження – 10072001.

*Console.Write($" Зчитування файлу: ");*

*string text = System.IO.File.ReadAllText(@"C:\Labs\Kripto\2\2\elga\elga\bin\Debug\net6.0\1.txt");*

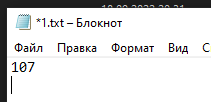


Рисунок 3 – початкове повідомлення

*List<BigInteger> array\_cipher\_text = new List<BigInteger>();*

*array\_cipher\_text = Cipher(text, p, y);*

*Console.WriteLine("Зашифроване повідомлення: ");*

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

*{*

*Console.WriteLine($"{i}:[{a}, {array\_cipher\_text[i]}] |--| ");*

*}*

*Console.WriteLine();*

*Console.Write($"\nДешифроване повідомлення: {text = Cipher\_RAZ(text.Length, array\_cipher\_text, x, p)}");*

*Console.ReadKey();*

*}*

*}*

*}*

Після цього виводиться шифроване повідомлення.

**

Рисунок 4 – шифроване повідомлення

І вже в кінці виводиться дешифроване повідомлення.

**

Рисунок 5 – дешифроване повідомлення

**Загальний код**

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*using System.Threading.Tasks;*

*using System.Numerics;*

*namespace ConsoleApp12*

*{*

*class Program*

*{*

*static int g\_main;*

*static BigInteger a;*

*/\*Definition number <g>\*/*

*public static bool Search\_g(int p, int g)*

*{*

*bool boolean = false;*

*List<BigInteger> array\_mod\_number = new List<BigInteger>();*

*BigInteger integer = ((BigInteger.Pow(g, 1)) % p);*

*array\_mod\_number.Add(integer);*

*for (int i = 2; i != p; i++)*

*{*

*integer = BigInteger.Pow(g, i) % p;*

*for (int j = 0; j != i - 1; j++)*

*{*

*if (array\_mod\_number[j] == integer)*

*{*

*g--;*

*array\_mod\_number.Clear();*

*i = 1;*

*integer = BigInteger.Pow(g, 1) % p;*

*array\_mod\_number.Add(integer);*

*break;*

*}*

*if ((j == i - 2) && (array\_mod\_number[j] != integer))*

*{*

*array\_mod\_number.Add(integer);*

*}*

*}*

*}*

*g\_main = g;*

*boolean = true;*

*return boolean;*

*}*

*/\*Definition simple number <p>\*/*

*public static int Search\_p()*

*{*

*Random random = new Random();*

*int p = 0;*

*Boolean boolean = false;*

*do*

*{*

*p = random.Next(601, 602);*

*for (int i = 2; i != p; i++)*

*{*

*if (i == p - 1)*

*{*

*boolean = Search\_g(p, p - 1);*

*break;*

*}*

*if (p % i == 0)*

*{*

*break;*

*}*

*}*

*}*

*while (boolean == false);*

*return p;*

*}*

*/\*Шифр\*/*

*public static List<BigInteger> Cipher(string text, int p, BigInteger y)*

*{*

*List<BigInteger> array = new List<BigInteger>();*

*Random random = new Random();*

*int k = random.Next(1, p - 1);*

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

*{*

*a = BigInteger.Pow(g\_main, k) % p;*

*array.Add((BigInteger.Pow(y, k) \* (int)text[i]) % p);*

*}*

*return array;*

*}*

*/\*Дешифрування\*/*

*public static string Cipher\_RAZ(int length\_text, List<BigInteger> array\_number, int x, int p)*

*{*

*string save\_text = "";*

*BigInteger integer;*

*for (int i = 0; i != length\_text; i++)*

*{*

*integer = (array\_number[i] \* (BigInteger.Pow(a, p - 1 - x))) % p;*

*save\_text += (char)integer;*

*}*

*return save\_text;*

*}*

*static void Main(string[] args)*

*{*

*int p = 0;*

*Random random = new Random();*

*p = Search\_p();*

*int x = 5; //закритий ключ*

*BigInteger y = BigInteger.Pow(g\_main, x) % p; //Відкритий ключ*

*Console.Write($"Числа [{p}, {g\_main}, {y}] ---> [p, g, y]\n");*

*Console.Write($"Вiдкритий ключ - это <y>\n");*

*Console.Write($"Закритий ключ: {x}\n");*

*Console.Write($" Зчитування файлу: ");*

*string text = System.IO.File.ReadAllText(@"C:\Labs\Kripto\2\2\elga\elga\bin\Debug\net6.0\1.txt");*

*List<BigInteger> array\_cipher\_text = new List<BigInteger>();*

*array\_cipher\_text = Cipher(text, p, y);*

*Console.WriteLine("Зашифроване повiдомлення: ");*

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

*{*

*Console.WriteLine($"{i}:[{a}, {array\_cipher\_text[i]}] |--| ");*

*}*

*Console.WriteLine();*

*Console.Write($"\nДешифроване повiдомлення: {text = Cipher\_RAZ(text.Length, array\_cipher\_text, x, p)}");*

*Console.ReadKey();*

*}*

*}*

*}*

**Висновок:** В результаті роботи було отримано програму в якій реалізовано програмне шифрування за методом Ель-Гамаля. Схему запропонував Тахер Ель-Гамаль 1985 року. Ель-Гамаль розробив один з варіантів алгоритму Діффі-Геллмана. Він удосконалив систему Діффі-Геллмана й отримав два алгоритми, які призначено для шифрування та для автентифікації. На відміну від RSA, алгоритм Ель-Гамаля не запатентовано, і тому він став дешевшою альтернативою, оскільки оплата внесків за ліцензію не потрібна. Вважається, що алгоритм потрапляє під дію патенту Діффі-Геллмана.

Нині криптосистеми з відкритим ключем вважаються найперспективнішими. До них належить і схема Ель-Гамаля, криптостійкість якої засновано на обчислювальній складності проблеми дискретного логарифмування, де за відомими p , g та y потрібно обчислити x , що задовольняє рівнянню: