Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы защиты информации

ОТЧЁТ

по лабораторной работе

на тему

Криптография с использованием эллиптических данных

Выполнил

Студент гр. 053502

Юрьев В. А.

Проверил

Ассистент кафедры информатики

Лещенко Е. А.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc148969133)

[2 Теоретические сведения 4](#_Toc148969134)

[3 Блок-схема алгоритма 5](#_Toc148969135)

[4 Демонстрация работы 6](#_Toc148969136)

[Выводы 7](#_Toc148969137)

[Приложение А (обязательное) Код программы 8](#_Toc148969138)

1. **ПОСТАНОВКА ЗАДАЧИ**

Реализовать схему шифрования (дешифрования) для аналога алгоритма

Эль-Гамаля на основе эллиптических кривых.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Шифрование алгоритмом Эль-Гамаля с использованием эллиптических кривых представляет собой криптографическую систему, которая обеспечивает конфиденциальность сообщений и основана на математической теории эллиптических кривых. Этот метод шифрования имеет множество практических применений, включая безопасную передачу данных в сети и создание цифровых подписей. Основные концепции и этапы шифрования с использованием алгоритма Эль-Гамаля на эллиптических кривых:

1. Выбор эллиптической кривой. Сначала выбирается эллиптическая кривая E над конечным полем Fp, где p — большое простое число. Эта кривая определяется уравнением в форме Вейерштрасса: E: y^2 = x^3 + ax + b, где a и b - коэффициенты, определенные в контексте конкретной кривой.
2. Генерация точки базы (G). Выбирается точка G на эллиптической кривой E вместе с её порядком n. Порядок n — это наименьшее положительное целое число, удовлетворяющее уравнению n\*G = O, где O — бесконечно удаленная точка на кривой.
3. Выбор секретного ключа (d). Случайным образом генерируется секретный ключ "a". Этот ключ используется для шифрования и подписи.
4. Вычисление открытого ключа (A). Открытый ключ A вычисляется как A = d\*G, где "d" - секретный ключ, а "G" — точка базы.
5. Шифрование сообщения. Перед отправкой сообщения, отправитель выбирает случайное целое число k и вычисляет точку C1 = k\*G и C2 = сообщение + k\*A. Точка C1 и C2 составляют шифртекст.
6. Расшифрование сообщения. Получатель использует свой секретный ключ "a" для вычисления обратной точки k = d\*C1, где k — случайное число, связанное с шифрованием. Затем получатель вычисляет исходное сообщение, вычитая точку k из C2: сообщение = C2 – k.

Алгоритм Эль-Гамаля на эллиптических кривых обеспечивает безопасность данных, так как даже зная открытый ключ A и шифртекст, вычисление секретного ключа "a" является вычислительно сложной задачей, основанной на дискретном логарифмировании на эллиптических кривых. Это делает метод безопасным для использования в практических приложениях криптографии.

1. **БЛОК-СХЕМА АЛГОРИТМА**

Схема алгоритма шифрования Эль-Гамаля приведена на рисунке 3.1.

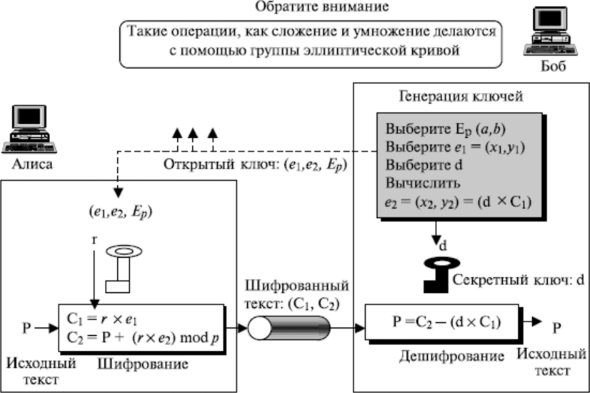


Рисунок 3.1 – Схема алгоритма шифрования Эль-Гамаля

1. **ДЕМОНСТРАЦИЯ РАБОТЫ**

Пример работы программы показан на рисунке 4.1.

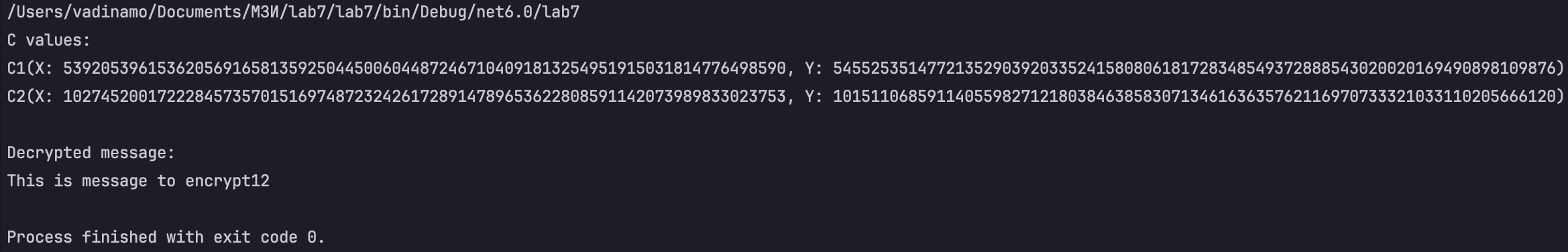


Рисунок 4.1 – Пример работы программы

**ВЫВОДЫ**

В ходе выполнения данной лабораторной работы была реализована схема шифрования (дешифрования) для аналога алгоритма Эль-Гамаля на основе эллиптических кривых.

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Код программы**

using System.Numerics;

using System.Text;

using lab7;

var message = File.ReadAllBytes("./input.txt");

var P = new EllipticCurvePoint(

BigInteger.Parse("2"),

BigInteger.Parse("4018974056539037503335449422937059775635739389905545080690979365213431566280"),

BigInteger.Parse("7"),

BigInteger.Parse("43308876546767276905765904595650931995942111794451039583252968842033849580414"),

BigInteger.Parse("57896044618658097711785492504343953926634992332820282019728792003956564821041")

);

var d = BigInteger.Parse("47296044618658097711785492524343953912234992332820282019728792003956564821041");

var Q = EllipticCurvePoint.Multiply(P, d);

var CValues = ElGamal.Encrypt(message, P, Q);

Console.WriteLine($"C values:\nC1(X: {CValues[0].x}, Y: {CValues[0].y})\nC2(X: {CValues[1].x}, Y: {CValues[1].y})\n");

Console.WriteLine($"Decrypted message:\n{Encoding.UTF8.GetString(ElGamal.Decrypt(CValues, d))}");

using System.Numerics;

namespace lab7;

public static class ElGamal

{

private static BigInteger GenerateRandomBigInteger(BigInteger N)

{

var random = new Random();

var bytes = N.ToByteArray();

BigInteger r;

do

{

random.NextBytes(bytes);

bytes[^1] &= 0x7F;

r = new BigInteger(bytes);

} while (r >= N);

return r;

}

private static EllipticCurvePoint GetPointFromBytes(byte[] messageBytes, EllipticCurvePoint P)

{

var pLength = P.p.ToByteArray().Length;

if (messageBytes.Length >= pLength - 2)

{

throw new Exception($"M({messageBytes.Length}) should be less than p (Max M Length = {pLength - 2} symbols)");

}

var message = messageBytes.Concat(new byte[P.p.ToByteArray().Length - messageBytes.Length]).ToArray();

message[messageBytes.Length] = 0xff;

return new EllipticCurvePoint(

new BigInteger(message),

0,

P.a,

P.b,

P.p

);

}

private static byte[] GetBytesFromPoint(EllipticCurvePoint P)

{

var messageBytes = P.x.ToByteArray();

return messageBytes.Take(Array.LastIndexOf(messageBytes,(byte)0xff)).ToArray();

}

public static EllipticCurvePoint[] Encrypt(byte[] messageBytes, EllipticCurvePoint P, EllipticCurvePoint Q)

{

var M = GetPointFromBytes(messageBytes, P);

var k = GenerateRandomBigInteger(P.p);

var C1 = EllipticCurvePoint.Multiply(P, k);

var C2 = M + EllipticCurvePoint.Multiply(Q, k);

return new[] { C1, C2 };

}

public static byte[] Decrypt(EllipticCurvePoint[] CValues, BigInteger d)

{

var temp = EllipticCurvePoint.Multiply(CValues[0], d);

temp.y = -temp.y;

var P = temp + CValues[1];

return GetBytesFromPoint(P);

}

}