Министерство образования Республики Беларусь Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ

И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №7

По теме «Криптография с использованием эллиптических кривых»

Выполнила: студентка гр. 053501 Шурко Т.А.

Проверил: ассистент кафедры информатики Лещенко Е. А.

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc150462052)

[1 Теоретические сведения 4](#_Toc150462053)

[1.1 Алгоритм Эль-Гамаля 4](#_Toc150462054)

[1.2 Блок-схема алгоритма Эль-Гамаля 4](#_Toc150462055)

[2 Результат выполнения задачи 6](#_Toc150462056)

[Заключение 7](#_Toc150462057)

[Приложение А 8](#_Toc150462058)

# ВВЕДЕНИЕ

2. Шифрование — обратимое преобразование информации в целях сокрытия от неавторизованных лиц, с предоставлением, в это же время, авторизованным пользователям доступа к ней. Главным образом, шифрование служит задачей соблюдения конфиденциальности передаваемой информации. Важной особенностью любого алгоритма шифрования является использование ключа, который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма.
3. Пользователи являются авторизованными, если они обладают определенным аутентичным ключом. Вся сложность и, собственно, задача шифрования состоит в том, как именно реализован этот процесс.
4. В целом, шифрование состоит из двух составляющих — зашифрование и расшифрование.
5. С помощью шифрования обеспечиваются три состояния безопасности информации. Первой является конфиденциальность: шифрование используется для скрытия информации от неавторизованных пользователей при передаче или при хранении. Второе — целостность. Шифрование используется для предотвращения изменения информации при передаче или хранении. Последнее это идентифицируемость. Шифрование используется для аутентификации источника информации и предотвращения отказа отправителя информации от того факта, что данные были отправлены именно им.
6. Для выполнения лабораторной работы 7 необходимо реализовать программное средство зашифрования и расшифрования данных из файла алгоритмом Эль-Гамаля.

# 1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

## 1.1 Алгоритм Эль-Гамаля

Схема Эль-Гамаля — [криптосистема](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) с открытым ключом, основанная на трудности вычисления [дискретных логарифмов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D1%8B%D0%B9_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC) в [конечном поле](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5). Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамаля лежит в основе бывших стандартов [электронной цифровой подписи](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C) в [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%A8%D0%90) ([DSA](https://ru.wikipedia.org/wiki/DSA)) и [России](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F) ([ГОСТ Р 34.10-94](https://ru.wikipedia.org/wiki/%D0%93%D0%9E%D0%A1%D0%A2_%D0%A0_34.10-94)).

## 1.2 Блок-схема алгоритма Эль-Гамаля

В данном подразделе на рисунке 1.1 изображена блок-схема алгоритма Эль-Гамаля, которая разделена на 2 основных блока: отправитель и получатель. В блоке отправитель сообщение преобразовывается и передается в виде кортежа из 2 значений. Получатель с использованием закрытого ключа x и пришедшей криптограммы производит расшифрование сообщения.

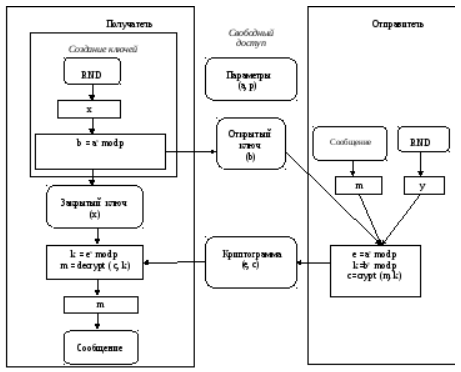


Рисунок 1.1 — Блок-схема алгоритма Эль-Гамаля

На рисунке 1.2 изображена схема алгоритма Эль-Гамаля в более наглядном виде.

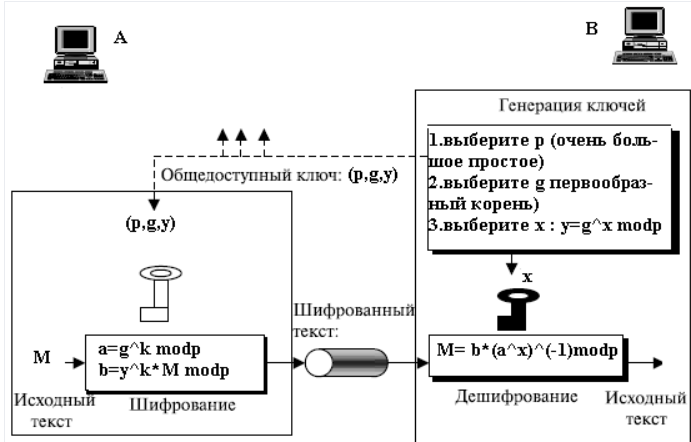


Рисунок 1.2 — Схема алгоритма Эль-Гамаля

# 2 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЗАДАЧИ

В результате выполнения задачи было реализовано зашифрование и расшифрование сообщения из текстового файла с помощью алгоритма Эль-Гамаля на языке программирования С#. Текст для шифрования необходимо поместить в файл и передать его наименование в виде первого параметра, результат применения алгоритма будет помещен в файл с наименованием, указанным во втором параметре. Текст исполняемой программы в приложении А.

Для запуска программного продукта необходимо подключить класс «Crypt.cs», а также указать файл, из которого берется текст и файл, в который будет помещаться результат работы (Рисунок 2.1). При запуске расшифрования параметры для запуска функции аналогичные.

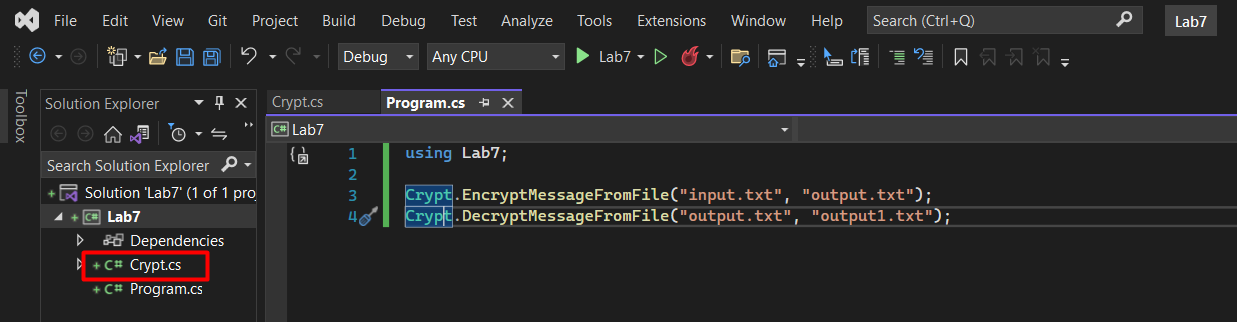


Рисунок 2.1 — Запускаемая программа

# ЗАКЛЮЧЕНИЕ

При выполнении лабораторной работы был изучен и реализован алгоритм шифрования и расшифрования Эль-Гамаля, рассмотрены и составлены блок-схемы алгоритмов, произведена их реализация на языке программирования C#, а также проверена работоспособность программного продукта.

Проблема дискретного логарифма состоит в том, что, зная основание степени и получившийся после возведения результат по модулю простого числа, невозможно за обозримое время определить, в какую именно степень было возведено основание. В схеме Эль-Гамаля потенциальный злоумышленник может получить значения a, p, (axmod p) и (aymod p). Однако из-за сложности определения чисел «х» и «y» у него не оказывается возможности вычислить значение k = (axymod p), которое так необходимо для прочтения шифровки.

По криптостойкости, в схеме Эль-Гамаля 512-битное число p приравнивается к 56-битному симметричному ключу, размер которого в настоящее время недостаточен для надежного шифрования. Поэтому на практике применяются р длиной в 768, 1024 и 1536 бит.

В отличие от RSA, алгоритм Эль-Гамаля не был запатентован и поэтому стал более дешёвой альтернативой, так как не требовалась оплата взносов за лицензию.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Код программы**

**Crypt.cs**

using System.Numerics;

namespace Lab7 {

public class Crypt {

private static Random rand = new Random();

//message

private static BigInteger m;

// open key

private static int p = 257;

private static BigInteger g = 3;

// close key

private static int x = rand.Next(1, p - 1);

private static int k = rand.Next(1, p - 1);

private static BigInteger y = BigInteger.ModPow(g, x, p);

public static byte[] Encrypt(byte[] message) {

m = new BigInteger(message, true);

BigInteger a = BigInteger.ModPow(g, k, p);

BigInteger temp = BigInteger.Pow(y, k) \* m;

BigInteger b = BigInteger.ModPow(temp, 1, p);

byte[] aByte = a.ToByteArray(true);

byte[] bByte = b.ToByteArray(true);

byte[] result = new byte[aByte.Length + bByte.Length];

Array.Copy(aByte, 0, result, 0, aByte.Length);

Array.Copy(bByte, 0, result, aByte.Length, bByte.Length);

return result;

}

public static byte Decrypt(byte[] encrypted) {

byte[] aByte = new byte[encrypted.Length / 2];

byte[] bByte = new byte[encrypted.Length / 2];

Array.Copy(encrypted, 0, aByte, 0, encrypted.Length / 2);

Array.Copy(encrypted, encrypted.Length / 2, bByte, 0, encrypted.Length / 2);

BigInteger a = new BigInteger(aByte, true);

BigInteger b = new BigInteger(bByte, true);

BigInteger temp = BigInteger.Pow(a, p - 1 - x);

BigInteger message = BigInteger.ModPow(b \* temp, 1, p);

byte[] messageByte = message.ToByteArray();

return messageByte[0];

}

public static void EncryptMessageFromFile(string inputFileName, string ouputFileName) {

using (FileStream outputFile = File.OpenWrite(ouputFileName)) {

using (FileStream inputFile = File.OpenRead(inputFileName)) {

using (BinaryReader binaryReader = new BinaryReader(inputFile)) {

using (BinaryWriter binaryWriter = new BinaryWriter(outputFile)) {

while (binaryReader.BaseStream.Position != binaryReader.BaseStream.Length) {

byte[] encryptedBytes = Encrypt(new byte[] { binaryReader.ReadByte() });

binaryWriter.Write(encryptedBytes);

}

}

}

}

}

}

public static void DecryptMessageFromFile(string inputFileName, string ouputFileName) {

using (FileStream outputFile = File.OpenWrite(ouputFileName)) {

using (FileStream inputFile = File.OpenRead(inputFileName)) {

using (BinaryReader binaryReader = new BinaryReader(inputFile)) {

using (BinaryWriter binaryWriter = new BinaryWriter(outputFile)) {

byte[] inputVal;

while ((inputVal = binaryReader.ReadBytes(2)).Length == 2) {

binaryWriter.Write(Decrypt(inputVal));

}

}

}

}

}

}

}

}

**Program.cs**

using Lab7;

Crypt.EncryptMessageFromFile("input.txt", "output.txt");

Crypt.DecryptMessageFromFile("output.txt", "output1.txt");