Министерство образования Республики Беларусь Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ

И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №3

По теме «Асимметричная криптография. Криптосистема Рабина»

Выполнила: студентка гр. 053501 Шурко Т.А.

Проверил: ассистент кафедры информатики Лещенко Е. А.

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc147869480)

[1 Теоретические сведения 4](#_Toc147869481)

[1.1 Криптосистема Рабина 4](#_Toc147869482)

[1.2 Блок-схема алгоритма 4](#_Toc147869483)

[2 Результат выполнения задачи 6](#_Toc147869484)

[Заключение 7](#_Toc147869485)

[Приложение А 8](#_Toc147869486)

# 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 3 необходимо реализовать программное средство зашифрования и расшифрования текстовых файлов используя асимметричную криптосистему Рабина.

## 1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

## 1.1 Криптосистема Рабина

Криптосистема Рабина представляет собой асимметричный криптографический метод, безопасность которого, как и у RSA связан со сложностью целочисленной факторизации. Однако криптосистема Рабина имеет то преимущество, что математически доказана ее вычислительная безопасность против атаки с выбранным открытым текстом до тех пор, пока злоумышленник не может эффективно разложить целые числа на множители, в то время как для RSA нет такого доказательства. Его недостаток состоит в том, что каждый выход функции Рабина может быть сгенерирован любым из четырех возможных входов; если каждый вывод представляет собой зашифрованный текст, при расшифровке требуется дополнительная сложность, чтобы определить, какой из четырех возможных входов был истинным открытым текстом.

## 1.2 Блок-схема алгоритма

В данном подразделе на рисунке 1.1 указана блок-схема зашифрования и расшифрования с помощью криптосистемы Рабина.

В блоке «Отправитель» в подблоке «Создание ключей» происходит формирование ключей, которые будут использоваться в алгоритме при зашифровании и расшифровании соответственно.

С помощью открытого ключа n происходит шифрование. Чтобы пользователю получить исходный текст, необходимо иметь закрытый ключ.

Закрытый ключ представляет из себя кортеж. Далее, используя эти ключи, осуществляются преобразования китайской теоремой об остатках и расширенным алгоритмом Евклида. В блоке «Получатель» показано как дешифруется сообщение.

Для решения проблемы 4 значений, для каждого шифруемого байта добавлялся ещё один байт, так называемая подпись. При расшифровании в каждом из 4 значений сверяется подпись. Если она эквивалентна той, которая использовалась при шифровании, то второй байт является исходным.

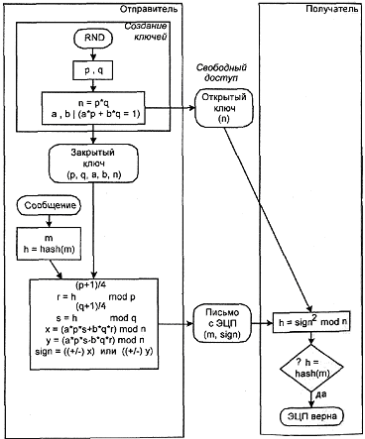


Рисунок 1.1 — Блок-схема криптосистемы Рабина

# 2 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЗАДАЧИ

В результате выполнения задачи было реализовано зашифрование и расшифрование текстовых файлов с помощью криптосистемы Рабина на языке программирования С#. Текст для шифрования необходимо поместить в файл и передать его наименование в виде первого параметра, текст после 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, а также указать файл, из которого берется текст и файл, в который будет помещаться результат работы (Рисунок 2.1). Для шифрования необходимо вызвать статический метод класса Crypt EncryptMessageFromFile. Аналогично, для дешифрования используется метод DecryptMessageFromFile.

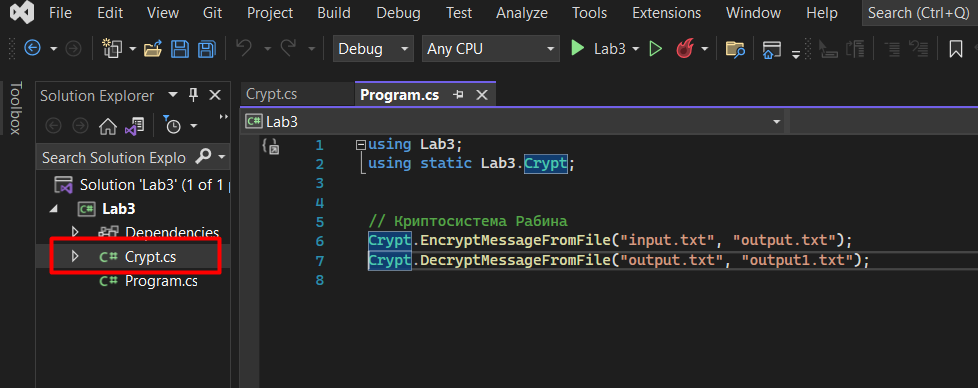


Рисунок 2.1 — Запускаемая программа

# ЗАКЛЮЧЕНИЕ

При выполнении лабораторной работы был изучен ассиметричный криптографический алгоритм Рабина, рассмотрены и составлены блок-схемы алгоритма, произведена его реализация на языке программирования C#, а также проверена работоспособность программного продукта.

Криптосистема, разработанная Рабином (Rabin), основана на сложности вычисления квадратного корня по модулю составного числа. Работа Рабина носила теоретический характер. В ней впервые приводилось доказательство стойкости криптосистем с открытым ключом: стойкость криптосистемы Рабина эквивалентна неразрешимости задачи о разложении целых чисел на множители. Алгоритм шифрования в криптосистеме Рабина чрезвычайно эффективен и пригоден для многих, практических приложений, например, для шифрования с помощью портативны устройств.

Недостаток состоит в том, что каждый выход функции Рабина может быть сгенерирован любым из четырех возможных входов; если каждый вывод представляет собой зашифрованный текст, при расшифровке требуется дополнительная сложность, чтобы определить, какой из четырех возможных входов был истинным открытым текстом. Данная проблема тоже имеет решения, например, использование дополнительной подписи при шифрования.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Код программы**

**Crypt.cs**

using System.Collections;

using System.Numerics;

namespace Lab3 {

internal class Crypt {

//Closed key

private static readonly int p = 107;

private static readonly int q = 223;

//Open key

private static readonly int n = p \* q;

//Decryption helper

private static readonly byte key = 0x80;

private static byte[] ConvertToByteArray(BitArray bits) {

byte[] bytes = new byte[2];

bits.CopyTo(bytes, 0);

return bytes;

}

private static List<BigInteger> ExtendedEuclideanAlgorithm(BigInteger r\_0, BigInteger r\_1,

BigInteger s\_0, BigInteger s\_1,

BigInteger t\_0, BigInteger t\_1) {

BigInteger q = r\_0 / r\_1;

BigInteger r\_next = r\_0 - r\_1 \* q;

BigInteger s\_next = s\_0 - s\_1 \* q;

BigInteger t\_next = t\_0 - t\_1 \* q;

if (r\_next == 0) {

return new List<BigInteger> () { s\_1, t\_1 };

}

return ExtendedEuclideanAlgorithm(r\_1, r\_next, s\_1, s\_next, t\_1, t\_next);

}

public static byte[] CryptByte(byte value) {

byte[] cryptedArr = new byte[2];

cryptedArr[1] = value;

cryptedArr[0] = key;

BitArray bitsArr = new BitArray(cryptedArr);

var reversedBitArr = new BitArray(bitsArr.Cast<bool>().Reverse().ToArray());

int[] array = new int[1];

reversedBitArr.CopyTo(array, 0);

int cryptedValue = array[0];

BigInteger val = cryptedValue;

BigInteger resultVal = BigInteger.ModPow(val, 2, n);

int res = (int)resultVal;

byte[] result = BitConverter.GetBytes(res);

return result;

}

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[] arr = CryptByte(binaryReader.ReadByte());

binaryWriter.Write(arr);

}

}

}

}

}

}

public static bool IsRightByte(BigInteger value, out byte byteRes) {

byteRes = 0x00;

int res = (int)value;

byte[] bytesVal = BitConverter.GetBytes(res);

byte[] cutRes = new byte[2];

Array.Copy(bytesVal, cutRes, 2);

BitArray bitsArr = new BitArray(cutRes);

BitArray reversedBitArr = new BitArray(bitsArr.Cast<bool>().Reverse().ToArray());

byte[] result = ConvertToByteArray(reversedBitArr);

if (result[0] == key) {

byteRes = result[1];

return true;

}

return false;

}

public static byte GetRightByte(BigInteger r\_1, BigInteger r\_2, BigInteger r\_3, BigInteger r\_4) {

byte decryptedByte = 0x00;

if (IsRightByte(r\_1, out decryptedByte)) {

return decryptedByte;

}

if (IsRightByte(r\_2, out decryptedByte)) {

return decryptedByte;

}

if (IsRightByte(r\_3, out decryptedByte)) {

return decryptedByte;

}

if (IsRightByte(r\_4, out decryptedByte)) {

return decryptedByte;

}

throw new Exception("Something went wrong while decrypting message!");

}

private static byte DectyptByte(byte[] inputVal) {

int value = BitConverter.ToInt32(inputVal, 0);

BigInteger m\_p = BigInteger.ModPow(value, (BigInteger)(0.25 \* (p + 1)), p);

BigInteger m\_q = BigInteger.ModPow(value, (BigInteger)(0.25 \* (q + 1)), q);

List<BigInteger> items = Crypt.ExtendedEuclideanAlgorithm(Math.Max(p, q), Math.Min(p, q), 1, 0, 0, 1);

BigInteger y\_p = 0, y\_q = 0;

if (Math.Max(p, q) == p) {

y\_p = items[0];

y\_q = items[1];

} else {

y\_p = items[1];

y\_q = items[0];

}

BigInteger r\_1 = (y\_p \* p \* m\_q + y\_q \* q \* m\_p) % n;

if (r\_1 < 0) {

r\_1 = n + r\_1;

}

BigInteger r\_2 = n - r\_1;

if (r\_2 < 0) {

r\_2 = n + r\_2;

}

BigInteger r\_3 = (y\_p \* p \* m\_q - y\_q \* q \* m\_p) % n;

if (r\_3 < 0) {

r\_3 = n + r\_3;

}

BigInteger r\_4 = n - r\_3;

if (r\_4 < 0) {

r\_4 = n + r\_4;

}

byte result = GetRightByte(r\_1, r\_2, r\_3, r\_4);

return result;

}

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(4)).Length == 4) {

binaryWriter.Write(DectyptByte(inputVal));

}

}

}

}

}

}

}

}

**Program.cs**

using Lab3;

using static Lab3.Crypt;

// Криптосистема Рабина

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

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