|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет» | |
|  | Факультет информационных технологий  Кафедра технологий программирования |
| Лабораторная работа №6 по курсу «Теория информации»  «Асимметричные криптосистемы. Алгоритм RSA» | |
| Выполнил | Студент гр. 21-ИТ-1  Макеёнок Д.И. |
| Проверила | Васильева Д.М. |
| Полоцк, 2023г. | |

**Ход работы**

Криптосистема RSA была предложена в 1977 году тремя учёными: Роном Ривестом, Ади Шамиром и Леонардом Адлеманом. Алгоритм шифрования данных RSA является асимметричным, поскольку для шифрования и расшифрования данных используются два различных ключа, называемых открытым и закрытым. Принципиальное отличие криптосистемы RSA от симметричных криптосистем заключается в том, что раскрытие ключа, которым было произведено шифрование, не влечёт за собой раскрытия исходного текста. Данный факт допускает пересылку ключа шифрования без использования каких-либо защищённых каналов связи. Это обеспечивается свойством криптосистемы RSA, в соответствии с которым использованный для шифрования данных открытый ключ не подходит для их расшифрования. Для расшифрования используется закрытый ключ, который не может быть получен из открытого.

**Задание:**

1. Разработать программное средство, выполняющее вычисление открытого ключа (K O) алгоритма RSA и побайтовое шифрование данным ключом по алгоритму RSA произвольного файла. Значения параметров p, q и K С , а также имя входного файла задаются пользователем. Программа должна осуществлять проверку ограничений на вводимые пользователем значения параметров алгоритма.

2. Разработать программное средство, выполняющее расшифрование файла, каждый 16-битный блок которого представляет собой зашифрованное по алгоритму RSA 8-битное значение. Значения модуля r и закрытого ключа K С задаются пользователем.

3. Разработать программное средство, выполняющее дешифрование (взлом) файла, каждый 16-битный блок которого представляет собой зашифрованное по алгоритму RSA 8-битное значение. Значения модуля r и открытого ключа K O, которым был зашифрован файл, задаются пользователем.

Листинг 1 - реализация Program.cs:

using Lab6;

bool IsPrime(int number)

{

if (number <= 1) return false;

if (number == 2) return true;

if (number % 2 == 0) return false;

var boundary = (int)Math.Floor(Math.Sqrt(number));

for (int i = 3; i <= boundary; i += 2)

if (number % i == 0)

return false;

return true;

}

var primes = RsaCrypt.SetPrimes(250);

foreach (var r in primes)

Console.Write(r + " ");

Console.WriteLine("Generating keys...");

var keys = RsaCrypt.SetKeys(primes);

Console.WriteLine($"public\_key:\t{keys.Item1}\n" +

$"private\_key:\t{keys.Item2}\n" +

$"n:\t{keys.Item3}\n");

Console.WriteLine("Please, input the file\_path: ");

var inputFilePath = Console.ReadLine();

while (inputFilePath == string.Empty)

inputFilePath = Console.ReadLine();

var inputMessage = File.ReadAllText(inputFilePath!);

var encodingList = RsaCrypt.Encoder(inputMessage, keys.Item1, keys.Item3);

Console.WriteLine("The encoded message: \n");

foreach (var k in encodingList)

Console.Write(k + " ");

var decodingText = RsaCrypt.Decoder(encodingList, keys.Item2, keys.Item3);

Console.WriteLine("\nThe decrypted text is: \n" +

decodingText);

while (true)

{

Console.WriteLine("Please, input the R key");

var key = Console.ReadLine();

switch (key)

{

case "R":

var tryCrackKeys = RsaCrypt.Crack(RsaCrypt.SetPrimes(250), keys.Item1, keys.Item3);

Console.WriteLine($"public\_key: {keys.Item1}\nprivate\_key: {tryCrackKeys.Item1}\nn: {tryCrackKeys.Item2}");

Console.WriteLine($"Decoded message: {RsaCrypt.Decoder(encodingList, tryCrackKeys.Item1, tryCrackKeys.Item2)}");

break;

}

}

Листинг 2 - реализация RsaCrypt.cs:

namespace Lab6;

public static class RsaCrypt

{

public static int gcd(int a, int b)

{

while (a != 0 && b != 0)

{

if (a > b)

a %= b;

else

b %= a;

}

return a | b;

}

public static List<int> SetPrimes(int length)

{

var boolMap = new bool[length];

boolMap[0] = true;

boolMap[1] = true;

for (var i = 2; i < length; i++)

for (var j = i \* 2; j < length; j += i)

boolMap[j] = true;

var prime = new List<int>();

for (var i = 0; i < boolMap.Length; i++)

if (!boolMap[i])

prime.Add(i);

return prime;

}

private static int PickRandomPrime(ref List<int> primeList)

{

var random = new Random();

if (primeList.Count == 0)

return 0;

var k = random.Next(0, primeList.Count);

var ret = primeList[k];

primeList.RemoveAt(k);

return ret;

}

private static int PickRandomPrime(List<int> primeList)

{

var random = new Random();

if (primeList.Count == 0)

return 0;

var k = random.Next(0, primeList.Count);

var ret = primeList[k];

return ret;

}

private static int Encrypt(double message, int publicKey, int n)

{

var e = publicKey;

var encryptedText = 1;

while (e-- > 0)

encryptedText = (int)(encryptedText \* message % n);

return encryptedText;

}

public static long Decrypt(int encryptedText, int privateKey, int n)

{

var d = privateKey;

var decrypted = 1;

while (d-- > 0)

decrypted = (decrypted \* encryptedText) % n;

return decrypted;

}

public static List<int> Encoder(string message, int publicKey, int n)

{

return message.Select(letter => Encrypt(letter, publicKey, n)).ToList();

}

public static string Decoder(List<int> encoded, int privateKey, int n)

{

var s = string.Empty;

foreach (var i in encoded)

s += (char)Decrypt(i, privateKey, n);

return s;

}

public static (int, int, int) SetKeys(List<int> primeList)

{

var prime1 = PickRandomPrime(ref primeList);

var prime2 = PickRandomPrime(ref primeList);

var n = prime1 \* prime2;

var fi = (prime1 - 1) \* (prime2 - 1);

var e = 2;

while (true)

{

if (gcd(e, fi) == 1)

break;

e++;

}

var publicKey = e;

var d = 2;

while (true)

{

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

break;

d++;

}

var privateKey = d;

return (publicKey, privateKey, n);

}

public static (int, int) Crack(List<int> primeList, int publicKey, int n)

{

var prime1 = PickRandomPrime(primeList);

var prime2 = PickRandomPrime(primeList);

while (n != prime1 \* prime2)

{

prime1 = PickRandomPrime(primeList);

prime2 = PickRandomPrime(primeList);

Console.WriteLine($"guessing the prime1: {prime1} and prime2: {prime2}");

}

var fi = (prime1 - 1) \* (prime2 - 1);

var d = 2;

while (true)

{

if (d \* publicKey % fi == 1 || d < 0)

break;

d++;

Console.WriteLine($"{d} current private\_key");

}

return (d, n);

}

}

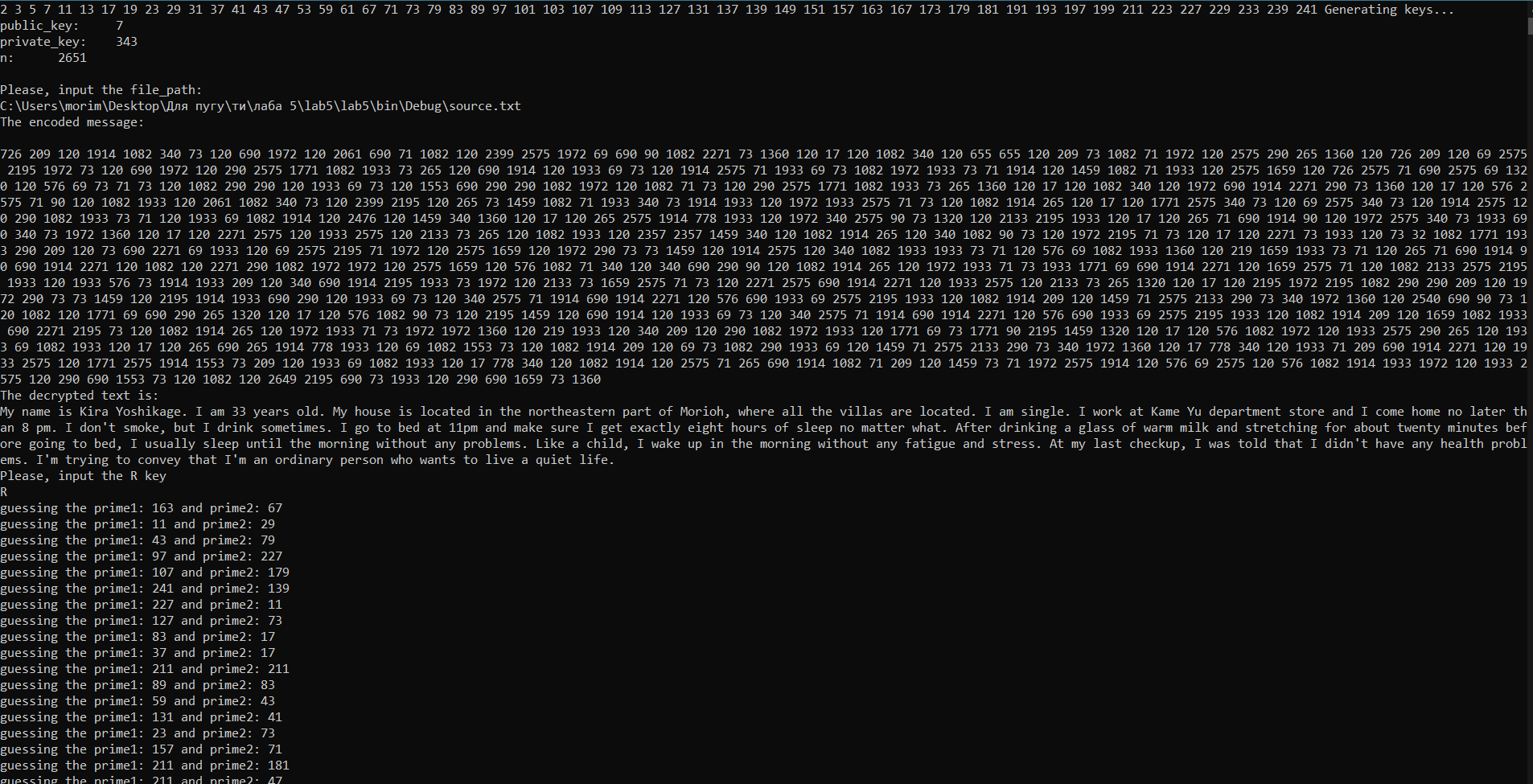


Рисунок 1. Результат работы программы

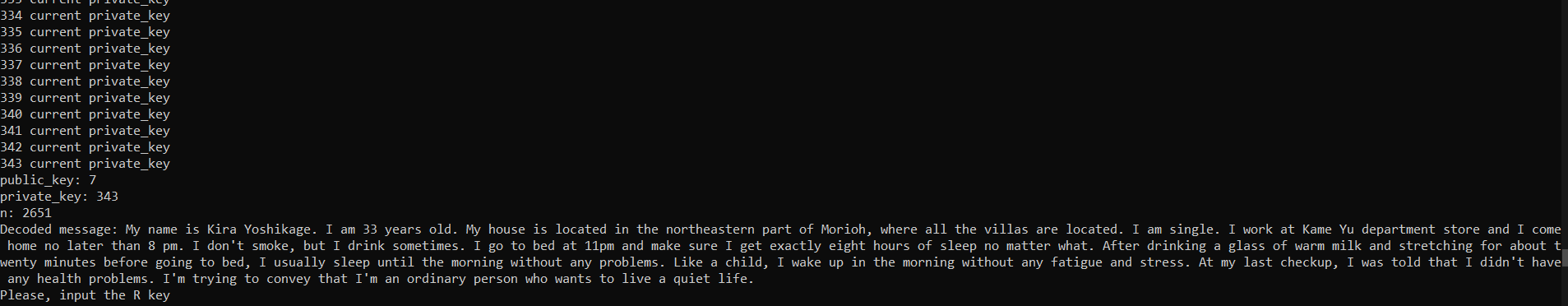


Рисунок 2. Расшифрованный текст