

***Федеральное агентство по рыболовству***

***Федеральное государственное бюджетное образовательное учреждение высшего образования***

***«Астраханский государственный технический университет»***

**Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована DQS по международному стандарту ISO 9001:2015**

**Институт информационных технологий и коммуникаций**

**Кафедра «Информационная безопасность»**

**Отчет по лабораторной работе №4. «Шифрование методом RSA»**

**по дисциплине**

**«КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ»**

**Работу выполнил**:

студент группы ДИБББ-31  
Чернов В.Ю. \_\_\_\_\_\_\_\_\_\_

**Работу проверил**:

Лим В.Г.\_\_\_\_\_\_\_\_\_\_

**АСТРАХАНЬ – 2023**

**Цель работы:** Изучить алгоритм шифрования RSA как представителя асимметричных криптосистем.

**Задание к лабораторной работе:** Написать на языке высокого уровня программу, которая бы реализовывала алгоритм шифрования/ расшифрования RSA

**Ход работы:**

Так как главный минус данного алгоритма – это то, что алгоритм основан на сложности факторизации больших целых чисел, то не каждый набор входных данных (p и q) может быть обработан стандартными средствами C#.

Возьму пример из методички и введу 3(p) и 11(q)(Рис.1)

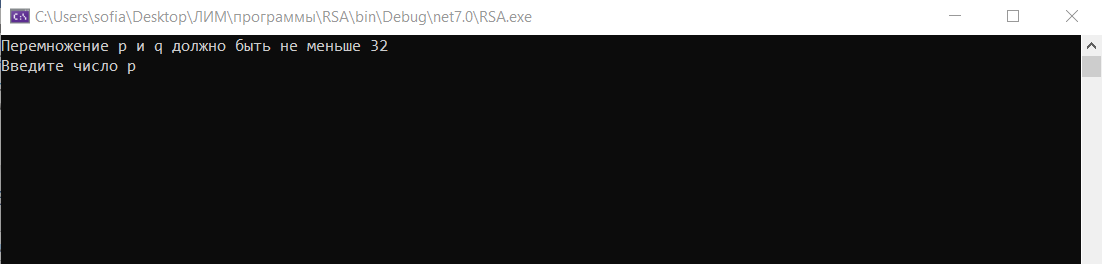


Рис.1

Получаем открытый ключ(e,n) и закрытый ключ(d,n) с помощью несложных вычислений:

int n = p \* q;

Console.WriteLine($"Число n = {n}");

int d = 2;

bool ObjieDeliteli = true;

do

{

d++;

if (d == 10)

{

Console.WriteLine("Не подобрано число d, попробуте другие входные данные");

Console.ReadKey();

System.Environment.Exit(0);

}

} while (ProverkaNaObjieDeliteli.CheckCommonDivisors(d, (p - 1) \* (q - 1)));

Console.WriteLine($"число d = {d} (не имеет общих делитеей с (p-1)\*(q-1) ) = {(p - 1) \* (q - 1)}");

int e = 0;

do

{

e++;

if (e == 10)

{

Console.WriteLine("Не подобрано число e, попробуте другие входные данные");

Console.ReadKey();

System.Environment.Exit(0);

}

} while ((e \* 3) % ((p - 1) \* (q - 1)) != 1);

Console.WriteLine($"число e = {e} (e \* 3 mod {(p - 1) \* (q - 1)} = 1)");

Console.WriteLine($"Открытый ключ (e,n) ({e},{n})");

Console.WriteLine($"Секретный ключ (d,n) ({d},{n})");

Число d и e ограничено до 10 включительно, потому что как правило выше этих значений не получится произвести расчеты стандартными средствами C#.

Получаем на выходе всю нужную нам информацию(Рис.2)

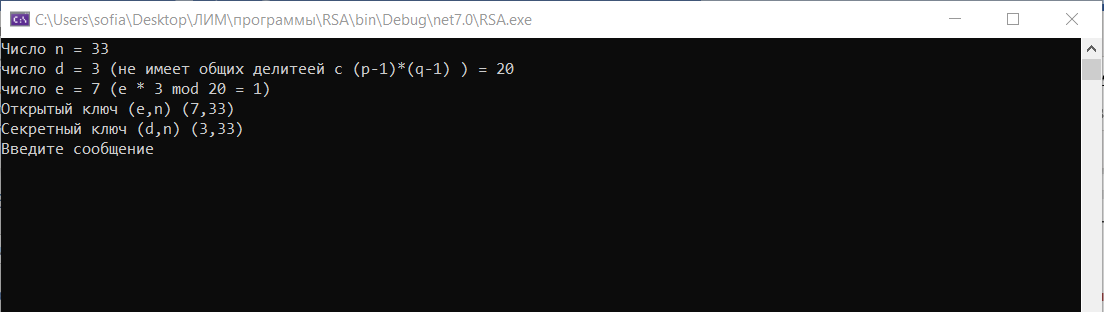


Рис.2

После ввода сообщения для шифрования («ШИФР») мы видим алгоритм шифрования каждой отдельной буквы с помощью открытого ключа и получившееся сообщение(Рис.3)

Далее мы видим алгоритм расшифровки каждой буквы с помощью закрытого ключа и получившееся сообщение(Рис.3)

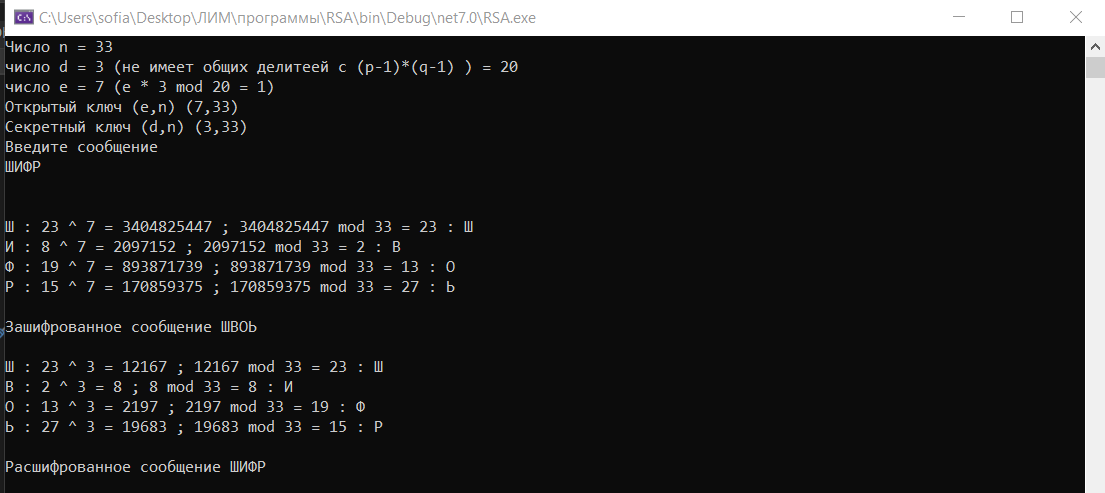


Рис.3

Программный код, отвечающий за весь алгоритм начиная с ввода сообщения для шифрования:

Console.WriteLine("Введите сообщение");

string message = Console.ReadLine();

int[] MessageToInt = new int[message.Length];

for (int i = 0; i < message.Length; i++)

{

MessageToInt[i] = alphabet.IndexOf(message[i]);

}

Console.WriteLine("\n");

int[] RSAMessageToInt = new int[MessageToInt.Length];

string RSAMessage;

StringBuilder BuffRSAMessage = new StringBuilder();

for (int i = 0; i < MessageToInt.Length; i++)

{

double buff = (Math.Pow(MessageToInt[i], e));

RSAMessageToInt[i] = Convert.ToInt32(Convert.ToInt64(buff) % n);

BuffRSAMessage.Append(alphabet[RSAMessageToInt[i]]);

Console.WriteLine($"{message[i]} : {MessageToInt[i]} ^ {e} = {buff} ; {buff} mod {n} = {RSAMessageToInt[i]} : {alphabet[RSAMessageToInt[i]]}");

}

RSAMessage = BuffRSAMessage.ToString();

Console.WriteLine($"\nЗашифрованное сообщение {RSAMessage}\n");

string DeRSAMessage;

StringBuilder BuffDeRSAMessage = new StringBuilder();

int[] DeRSAMessageToInt = new int[RSAMessageToInt.Length];

for(int i = 0; i < RSAMessageToInt.Length; i++)

{

double buff = (Math.Pow(RSAMessageToInt[i], d));

DeRSAMessageToInt[i] = (int)buff % n;

BuffDeRSAMessage.Append(alphabet[DeRSAMessageToInt[i]]);

Console.WriteLine($"{RSAMessage[i]} : {RSAMessageToInt[i]} ^ {d} = {buff} ; {buff} mod {n} = {DeRSAMessageToInt[i]} : {alphabet[DeRSAMessageToInt[i]]}");

}

DeRSAMessage = BuffDeRSAMessage.ToString();

Console.WriteLine($"\nРасшифрованное сообщение {DeRSAMessage}");

Console.ReadLine();

**Контрольные вопросы**

1. **Основа алгоритма.**

* RSA основан на сложности факторизации больших целых чисел.
* Он использует два простых числа, называемых "минсами" (или "простыми множителями"), для генерации ключей и шифрования/дешифрования данных.

1. **Описание алгоритма.**

**Генерация ключей:**

* Выбираются два различных простых числа p и q.
* Вычисляется их произведение n = p \* q, которое является модулем для шифрования и дешифрования.
* Вычисляется значение функции Эйлера от n, обозначаемое как φ(n).
* Выбирается целое число e, взаимно простое с φ(n), в качестве открытого экспонента.
* Вычисляется число d, обратное по модулю φ(n) к e, в качестве закрытого экспонента.
* Пара (n, e) составляет открытый ключ, а пара (n, d) - закрытый ключ.

**Шифрование данных:**

* Представляется сообщение в виде числа m, меньшего чем n.
* Вычисляется зашифрованное сообщение c = m^e mod n.

**Дешифрование данных:**

* Полученное зашифрованное сообщение c используется для вычисления исходного сообщения m = c^d mod n.

1. **Преимущества и недостатки.**

**Преимущества:**

* Криптографическая безопасность: RSA обеспечивает высокий уровень безопасности при правильной реализации и использовании достаточно больших ключей.
* Асимметричность: RSA использует открытый и закрытый ключи, что позволяет безопасно обмениваться данными без предварительного обмена секретным ключом.
* Широкое применение: RSA широко используется для шифрования данных, цифровой подписи и других криптографических протоколов.

**Недостатки:**

* Вычислительная сложность: RSA требует выполнения операций с большими целыми числами, что может быть затратным с вычислительной точки зрения.
* Ограничение длины сообщения: RSA может шифровать только сообщения, которые меньше по размеру, чем модуль n. Для шифрования более длинных сообщений требуется использование гибридных схем шифрования.
* Зависимость от безопасности ключей: Безопасность RSA полностью зависит от безопасности закрытого ключа. Компрометация закрытого ключа может привести к нарушению безопасности системы.

**Вывод:** Я изучил алгоритм шифрования RSA как представителя асимметричных криптосистем.