Министерство образования Республики Беларусь

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет к лабораторной работе № 7

«Исследование асимметричных шифров RSA и Эль-Гамаля»

Выполнила:

студентка 3 курса 2 группы

Шастовская М. С.

Вариант: 12

Преподаватель:

Сазонова Д.В.

Минск 2023

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамаля.

**Задачи:**

* Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров RSA и Эль-Гамаля.
* Разработать приложение для реализации асимметричного зашифрования/расшифрования на основе алгоритмов RSA и Эль-Гамаля.
* Выполнить анализ криптостойкости асимметричных шифров RSA и Эль-Гамаля.
* Оценить скорость зашифрования/расшифрования реализованных шифров.
* Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

1 Теоретические сведения

*Теорема 1: основная теорема арифметики.* Всякое натуральное число N, кроме 1, можно представить как произведение простых множителей:

N = p1p2p3...pz, z > 1.

Определение 1. *Задача дискретного логарифмирования* формулируется так: для данных целых чисел а и b, 1 < а, b < n, найти логарифм – такое целое число х, что

ax ≡ b (mod n),

если такое число существует.

По аналогии с вещественными числами используется обозначение х = logab.

*Теорема 2: китайская теорема об остатках.* В общем случае если разложение числа N на простые множители представляет собой p1p2…pt (некоторые простые числа могут встречаться несколько раз), то система уравнений

(x mod pi) ≡ ai,

где i = 1, 2, …, t, имеет единственное решение: x, меньшее N.

Безопасность RSA основана на трудности разложения на множители больших чисел. Открытый и закрытый ключи являются функциями двух больших простых чисел.

Для генерации двух ключей: тайного и открытого (а по сути – двух взаимосвязанных частей одного ключа, т. е. ключа, принадлежащего одному физическому лицу (или группе лиц), либо одному юридическому лицу), используются два больших случайных простых числа p и q. Для максимальной большей криптостойкости нужно выбирать p и q равной длины. Рассчитывается произведение: n = pq. Это есть один из трех компонент ключа, состоящего из чисел n, e, d.

Затем случайным образом выбирается второй компонент ключа (открытый ключ или ключ зашифрования, e, такой что e и (p – 1)(q – 1) являются взаимно простыми числами; вспомним, что (p – 1)(q – 1) = φ(n) – функция Эйлера). Б. Шнайер рекомендует число е выбирать из ряда: 3, 17, 216 + 1.

Наконец, расширенный алгоритм Евклида используется для вычисления третьего компонента ключа: ключа расшифрования d такого, что выполняется условие:

ed ≡ 1 (mod φ(n)).

Зашифрование. Если шифруется сообщение М, состоящее из r блоков: m1, m2, …, mi, …, mr, то шифртекст С будет состоять из такого же числа (r) блоков, представляемых числами:

ci ≡ (mi)e mod n.

Расшифрование. Для расшифрования каждого зашифрованного блока производится вычисление вида:

mi ≡ (ci) d mod n.

Алгоритм Эль-Гамаля отличается от алгоритма RSA несколькими параметрами и особенностями:

1) генерацией ключевой информации и числом компонент, составляющих ключ;

2) каждому блоку (символу) открытого сообщения в шифртексте на основе алгоритма Эль-Гамаля соответствуют 2 блока (в RSA – один-один);

3) в алгоритме Эль-Гамаля при зашифровании используется число (обозначим его k), которое практически никак не связано с ключевой информацией получателя и которое принимает (по определению) различные значения при зашифровании различных блоков сообщения.

Генерация ключевой информации. Выбирается простое число р. Выбирается число (g, g < p), являющееся первообразным корнем числа р – очень важный элемент с точки зрения безопасности алгоритма.

Далее выбирается число х (х < p) и вычисляется последний компонент ключевой информации:

y ≡ gх mod р.

Владельцу сформированной ключевой информации, состоящей из 4 чисел, может посылаться некоторый шифртекст, созданный с использованием открытого ключа получателя: p, g, y. Расшифрование шифртекста получатель производит своим тайным ключом: p, g, х.

*Определение 2.* Первообразный корень (primary (residual) root) по модулю р является таким числом, что его степени (gi , 1 ≤ i ≤ p – 1) дают все возможные по модулю р вычеты (остатки), которые взаимно просты с p.

*Зашифрование сообщения*. Как ранее, предположим, что сообщение М = {mi}, где mi – i-й блок сообщения.

*Расшифрование ci.* Выполняется по следующей формуле:

mi ≡ (bi(ai) x ) –1) mod p

или

mi ≡ (bi(ai) р – x – 1) mod p,

где (ax ) –1 – обратное значение числа ax по модулю p.

2 Практическая часть

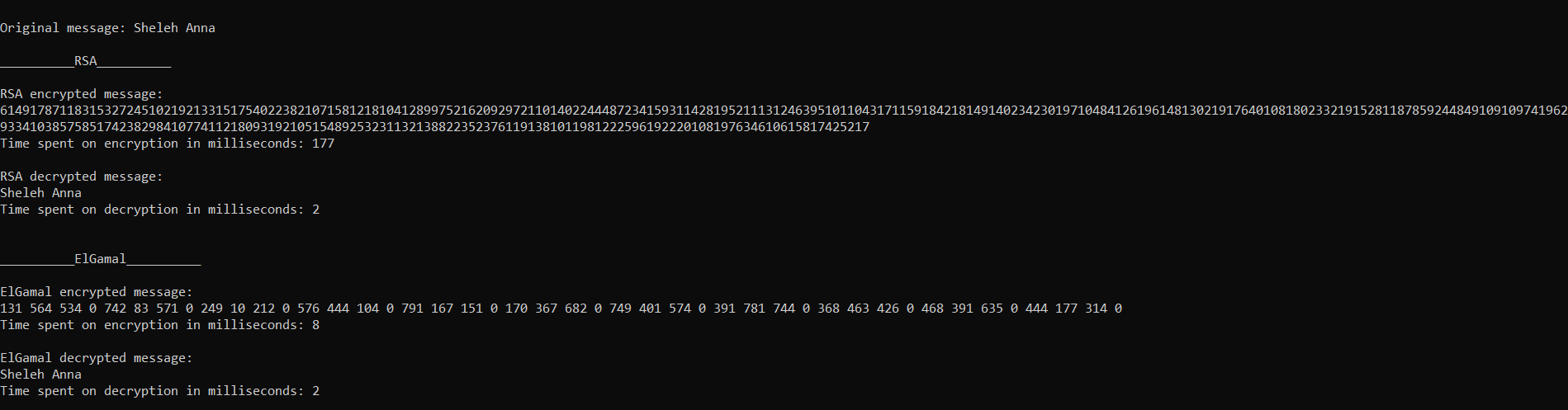


Рисунок 2.1 – Результат шифрования/дешифрования исходного сообщения шифрами RSA и Эль-Гамаля

На данных скриншотах отображен процесс работы программы с шифрованием и дешифрованием исходного сообщения шифрами RSA и Эль-Гамаля, а также определение времени выполнения операций.

Основные функции программы представлены на рисунке 2.2.

public static void Task1()

{

int a = 20;

System.Numerics.BigInteger[] xarr = { System.Numerics.BigInteger.Pow(10, 20),

System.Numerics.BigInteger.Pow(10, 40),

System.Numerics.BigInteger.Pow(10, 60),

System.Numerics.BigInteger.Pow(10, 80),

System.Numerics.BigInteger.Pow(10, 100)};

var size1 = 1024;

var size2 = 2048;

var r1 = new Random();

var r2 = new Random();

BigMath.BigInteger n1 = new BigMath.BigInteger(size1, r1);

BigMath.BigInteger n2 = new BigMath.BigInteger(size2, r2);

CalculateBigIntegers(a, xarr, n1);

CalculateBigIntegers(a, xarr, n2);

}

public static string Encrypt(string value, string pathToFile)

{

string result = "";

List<BigInteger> cipherASCIIElGamal = EncryptMethod(Encoding.Unicode.GetBytes(value));

File.WriteAllText(pathToFile, Convert.ToString(x));

foreach (long l in cipherASCIIElGamal)

{

result += Convert.ToString(l);

result += " ";

}

return result;

}

public static string Decrypt(string value, string pathToFile)

{

string[] words = value.Split(new char[] { ' ' });

List<BigInteger> cipherASCIIElGamal = new List<BigInteger>();

for (int i = 0; i < words.Length - 1; i++)

{

BigInteger helper = BigInteger.Parse(words[i]);

cipherASCIIElGamal.Add(helper);

}

return Encoding.Unicode.GetString(DecryptMethod(cipherASCIIElGamal, pathToFile));

}

public static byte[] RSAEncrypt(byte[] DataToEncrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)

{

try

{

byte[] encryptedData;

//Create a new instance of RSACryptoServiceProvider.

using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())

{

//Import the RSA Key information. This only needs

//toinclude the public key information.

RSA.ImportParameters(RSAKeyInfo);

//Encrypt the passed byte array and specify OAEP padding.

//OAEP padding is only available on Microsoft Windows XP or

//later.

encryptedData = RSA.Encrypt(DataToEncrypt, DoOAEPPadding);

}

return encryptedData;

}

//Catch and display a CryptographicException

//to the console.

catch (CryptographicException e)

{

Console.WriteLine(e.Message);

return null;

}

}

public static byte[] RSADecrypt(byte[] DataToDecrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)

{

try

{

byte[] decryptedData;

//Create a new instance of RSACryptoServiceProvider.

using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())

{

//Import the RSA Key information. This needs

//to include the private key information.

RSA.ImportParameters(RSAKeyInfo);

//Decrypt the passed byte array and specify OAEP padding.

//OAEP padding is only available on Microsoft Windows XP or

//later.

decryptedData = RSA.Decrypt(DataToDecrypt, DoOAEPPadding);

}

return decryptedData;

}

//Catch and display a CryptographicException

//to the console.

catch (CryptographicException e)

{

Console.WriteLine(e.ToString());

return null;

}

}

Рисунок 2.2 – Основные функции реализации.

Вывод

В ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложений для реализации асимметричны шифров RSA и Эль-Гамаля.