Министерство образования Республики Беларусь

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет к лабораторной работе № 6

«Исследование асимметричных шифров»

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 ассиместричных шифров.

**Задачи:**

* Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров.
* Разработать приложение для реализации указанных преподавателем методов генерации ключевой информации и ее использования для асимметричного зашифрования/расшифрования.
* Выполнить анализ криптостойкости асимметричных шифров.
* 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

В основу асимметричной криптографии положена идея использовать ключи парами: один – для зашифрования (открытый, или публичный, ключ), другой – для расшифрования (тайный ключ). Отметим, что указанная пара ключей принадлежит получателю зашифрованного сообщения. Все алгоритмы шифрования с открытым ключом основаны на использовании односторонних функций, к числу которых, как известно, относится вычисление дискретного логарифма.

*Односторонней функцией (one-way function) называется математическая функция, которую относительно легко вычислить, но трудно найти по значению функции соответствующее значение аргумента, т. е. зная х, легко вычислить f(x), но по известному f(x) трудно найти подходящее значение x.*

Алгоритмы шифрования с открытым ключом можно использовать для решения следующих задач:

* зашифрования/расшифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа; • формирования цифровой подписи под электронными документами;
* распределения секретных ключей, используемых далее при шифровании документов симметричными методами.

По мнению Диффи и Хеллмана, алгоритм шифрования с открытым ключом должен:

* вычислительно легко создавать пару (открытый ключ e – закрытый ключ d);
* вычислительно легко зашифровывать сообщение Mi открытым ключом; • вычислительно легко расшифровывать сообщение Ci, используя закрытый ключ;
* обеспечивать непреодолимую вычислительную сложность определения соответствующего закрытого ключа при известном открытом ключе;
* обеспечивать непреодолимую вычислительную сложность восстановления исходного (открытого сообщения Mi) зная только открытый ключ и зашифрованное сообщение Ci.

*Ранцевый (рюкзачный) вектор S = (s1, ..., sz) – это упорядоченный набор из z, z ≥ 3, различных натуральных чисел si. Входом задачи о ранце (рюкзаке) называем пару (S, S), где S – рюкзачный вектор, а S – натуральное число.*

Решением для входа (S, S) будет такое подмножество из S, сумма элементов которого равняется S.

Суть метода для шифрования состоит в том, что существуют две различные задачи укладки ранца: одна из них решается легко и характеризуется линейным ростом трудоемкости, а другая решается трудно. Легкий для укладки ранец можно трансформировать в трудный.

*Сверхвозрастающей называется последовательность, в которой каждый последующий член больше суммы всех предыдущих.*

Алгоритм укладки ранца на основе сверхвозрастающей последовательности:

1. В качестве текущего выбирается число S, которое сравнивается с «весом» самого тяжелого предмета (dz); если текущий вес меньше веса данного предмета, то его в ранец не кладут (0), в противном случае его укладывают (1) в ранец и переходят к анализу очередного (в общем случае – i-го предмета).

2. Если на предыдущем (i-м шаге) предмет пополнил ранец, то текущий вес уменьшают на вес положенного предмета (S = S – di); переходят к следующему по весу предмету в последовательности: di – 1.

Шаги повторяются до тех пор, пока процесс не закончится. Если текущий вес уменьшится до нуля (S = 0), то решение найдено. В противном случае – нет.

Для зашифрования сообщения (М) оно сначала разбивается на блоки, по размерам равные числу (z) элементов последовательности в ранце. Затем, считая, что 1 указывает на присутствие элемента последовательности в ранце, а 0 – на его отсутствие, вычисляются полные веса рюкзаков (Si, i = 1, …, z): по одному ранцу для каждого блока сообщения с использованием открытого ключа получателя e.

Для расшифрования сообщения получатель (используя свой тайный ключ d: сверхвозрастающую последовательность) должен сначала определить такое обратное к а число а–1, что аа–1 mod n ≡ 1.

Ранцевые криптосистемы не являются криптостойкими. А. Шамир и Р. Циппел обнаружили, что зная числа а, a–1 и n («секретную лазейку»), можно восстановить сверхвозрастающую последовательность по нормальной последовательности.

2 Практическая часть

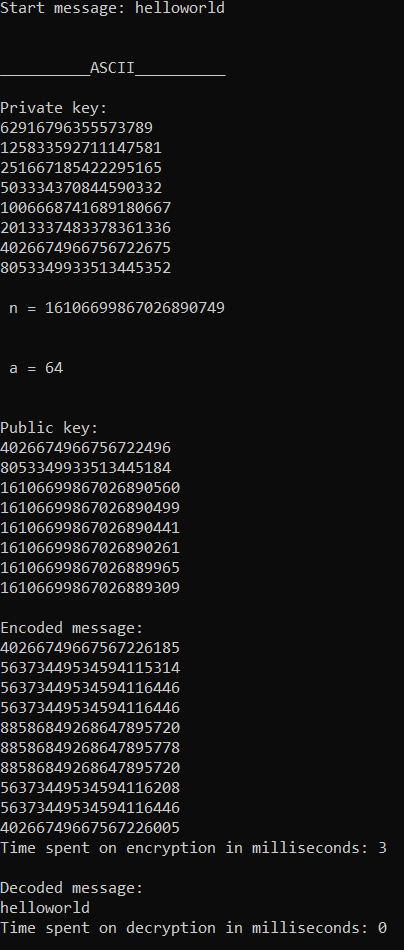


Рисунок 2.1 – Результат шифрования/дешифрования исходного сообщения с помощью кодировочной таблицы ASCII.

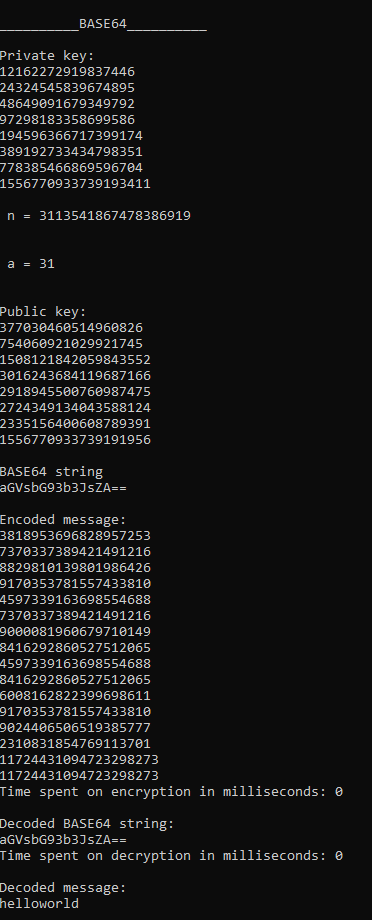


Рисунок 2.2 – Результат шифрования/дешифрования исходного сообщения с помощью кодировочной таблицы Base4.

На данных скриншотах отображен процесс работы программы с шифрованием и дешифрованием исходного сообщения кодировочными таблицами Base64, ASCII, а также вычисление открытого ключа, оценка выполнения операций зашифрования и расшифрования.

Основные функции программы представлены на рисунке 2.3.

static List<BigInteger> EncodeMessage(string message, List<BigInteger> publicKeyList)

{

List<Byte> Bytes = ascii.GetBytes(message).ToList();

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

foreach (var item in Bytes)

{

string symbol = Convert.ToString(item, 2);

BigInteger weight = 0;

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

{

if (symbol[symbol.Length - 1 - i] == '1')

weight += publicKeyList[publicKeyList.Count - 1 - i];

}

encodedMessageASCII.Add(weight);

}

return encodedMessageASCII;

}

static string DecodeMessage(List<BigInteger> encodedMessage, List<BigInteger> privateKeyList, BigInteger a\_inverse, BigInteger n)

{

List<string> symbols = new List<string>();

for (int i = 0; i < encodedMessage.Count; i++)

{

string symbol = "";

BigInteger decodedCode = (encodedMessage[i] \* a\_inverse) % n;

for (int j = 0; decodedCode != 0; j++)

{

if (decodedCode - privateKeyList[privateKeyList.Count - 1 - j] >= 0)

{

decodedCode -= privateKeyList[privateKeyList.Count - 1 - j];

symbol = '1' + symbol;

}

else

{

symbol = '0' + symbol;

}

}

symbol = symbol.PadLeft(8, '0');

symbols.Add(symbol);

}

List<byte> symbolsCodes = new List<byte>();

for (int i = 0; i < symbols.Count; i++)

{

symbolsCodes.Add((byte)GetIntFromString(symbols[i]));

}

string decodedMessage = "";

foreach (var item in ascii.GetChars(symbolsCodes.ToArray()))

{

decodedMessage += item;

}

return decodedMessage;

}

static List<BigInteger> EncodeMessageBase64(string message, List<BigInteger> publicKeyList)

{

List<Byte> Bytes = Encoding.UTF8.GetBytes(message).ToList();

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

foreach (var item in Bytes)

{

string symbol = Convert.ToString(item, 2);

BigInteger weight = 0;

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

{

if (symbol[symbol.Length - 1 - i] == '1')

weight += publicKeyList[publicKeyList.Count - 1 - i];

}

encodedMessageASCII.Add(weight);

}

return encodedMessageASCII;

}

static string DecodeMessageBase64(List<BigInteger> encodedMessage, List<BigInteger> privateKeyList, BigInteger a\_inverse, BigInteger n)

{

List<string> symbols = new List<string>();

for (int i = 0; i < encodedMessage.Count; i++)

{

string symbol = "";

BigInteger decodedCode = (encodedMessage[i] \* a\_inverse) % n;

for (int j = 0; decodedCode != 0; j++)

{

if (decodedCode - privateKeyList[privateKeyList.Count - 1 - j] >= 0)

{

decodedCode -= privateKeyList[privateKeyList.Count - 1 - j];

symbol = '1' + symbol;

}

else

{

symbol = '0' + symbol;

}

}

symbol = symbol.PadLeft(8, '0');

symbols.Add(symbol);

}

List<byte> symbolsCodes = new List<byte>();

for (int i = 0; i < symbols.Count; i++)

{

symbolsCodes.Add((byte)GetIntFromString(symbols[i]));

}

string decodedMessage = "";

foreach (var item in Encoding.UTF8.GetChars(symbolsCodes.ToArray()))

{

decodedMessage += item;

}

return decodedMessage;

}

Рисунок 2.3 – Основные функции реализации.

Вывод

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