МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9 НА ТЕМУ:**

**ИССЛЕДОВАНИЕ АСИММЕТРИЧНЫХ ШИФРОВ**

                                                                 Выполнил студент 3 курса 4 группы

Чёрная Яна Руслановна

Минск 2025

# Теоретические сведения.

В основу асимметричной криптографии положена идея использовать ключи парами: один – для зашифрования Исследование ассиметричных шифров 103 (открытый, или публичный, ключ), другой – для расшифро вания (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 относи тельно легко вычислить, но трудно найти по значению функции соответствующее значение аргумента, т. е. зная х, легко вычис лить f(x), но по известному f(x) трудно найти подходящее значе ние x.

Практически первой реализацией идеи Диффи – Хеллмана стал алгоритм согласования по открытому каналу тайного ключа между абонентами А и В.

Суть метода для шифрования состоит в том, что суще ствуют две различные задачи укладки ранца: одна из них решается легко и характеризуется линейным ростом трудо емкости, а другая решается трудно. Легкий для укладки ранец можно трансформировать в трудный. Трудный для укладки ранец применяется в качестве откры того ключа, который легко использовать для зашифрования, но невозможно – для расшифрования. В качестве закрытого ключа применяется легкий для укладки ранец, который предоставляет простой способ расшифрования сообщения.

# Задание 1.

Для начала, в качестве открытого ключа, необходимо было сгенерировать сверхвозрастающую последовательность. Сверхвозрастающая последовательность, это такая последовательность, в которой каждый следующий её член больше суммы предыдущих. В качестве закрытого ключа *d* (легкого для укладки ранца) используется последовательность, состоящая из *z* элементов: *d1*, *d2*, …, *dz*: *d* = {*di*}, *i* = 1, …, *z*. Код функции, генерирующею данную последовательность, представлен в листинге 1.1.

public static List<int> GenerateSuperIncreasingSequence(int length)

{

List<int> sequence = new List<int>();

int total = 0;

for (int i = 0; i < length; i++)

{

int nextValue = total + random.Next(1, 10);

sequence.Add(nextValue);

total += nextValue;

}

return sequence;

}

Листинг 1.1 – Функция, генерирующая сверхвозрастающую последовательность

Затем необходимо было высчитать нормальную последовательность, которая должна использоваться в качестве закрытого ключа. Она высчитывается по вот такой формуле:

*ei* ≡ *dia* mod *n*

Значение модуля *n* должно быть больше суммы всех чисел последовательности; кроме того, НОД (*а*, *n*) = 1. Код функции, выстраивающей нормальную последовательность, представлен в листинге 1.2.

public static List<int> GenerateNormalSequence(List<int> superIncreasingSequence, int q, int r)

{

return superIncreasingSequence.Select(x => (x \* r) % q).ToList();

}

Листинг 1.2 – Функция, высчитывающая нормальную последовательность

Далее, для шифрования, сообщение разбиваем на блоки, размер которых равен *z* битов. 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*. Алгоритм шифрования реализован в функции Encrypt. Код функции представлен в листинге 1.3.

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

{

byte[] bytes = Encoding.ASCII.GetBytes(message);

List<int> encrypted = new List<int>();

foreach (byte b in bytes)

{

int sum = 0;

for (int i = 0; i < 8; i++)

{

if ((b & (1 << i)) != 0)

{

sum += publicKey[i];

}

}

encrypted.Add(sum);

}

return encrypted;

}

Листинг 1.3 – Функция шифрования рюкзаком

Для расшифрования сообщения получатель должен сначала определить такое обратное к *а* число *а–1*, что

*аа*–1 mod *n* ≡ 1

После определения обратного числа каждое значение шифрограммы (*ci*) преобразуется в соответствии со следующим соотношением:

*Si* ≡ *ciа*–1 mod *n*

Код функции расшифрования представлен в листинге 1.4.

public static string Decrypt(List<int> encryptedMessage, List<int> privateKey, int q, int r)

{

int rInverse = ModInverse(r, q);

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

foreach (int encryptedValue in encryptedMessage)

{ int cPrime = (encryptedValue \* rInverse) % q; byte b = 0;

for (int i = privateKey.Count - 1; i >= 0; i--)

{

if (cPrime >= privateKey[i])

{ cPrime -= privateKey[i]; b |= (byte)(1 << i);}

} decryptedBytes.Add(b);

}return Encoding.ASCII.GetString(decryptedBytes.ToArray());

}

Листинг 1.4 – Функция расшифрования рюкзаком

Результат выполнения шифрования и расшифрования ранцем представлен на рисунке 1.1.

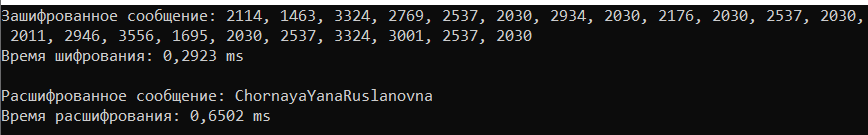


Рисунок 1.1 – Результат выполнения шифрования и расшифрования ранцем

# Задание 2.

Для измерения скорости выполнения алгоритмов шифрования и расшифрования, использовался текст с длиной 1905 символов. Результаты шифрования и расшифрования текста в ASCII представлен на рисунке 2.1 и текста в кодировке Base64 на рисунке 2.2.

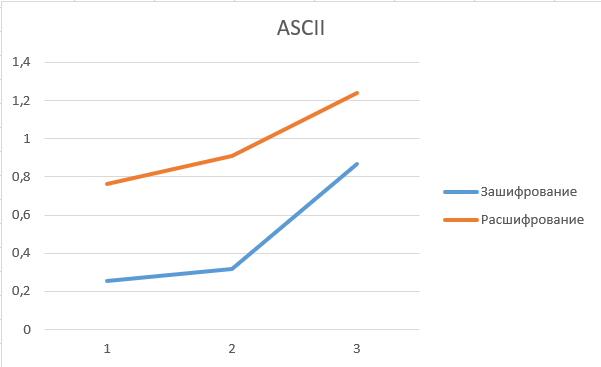


Рисунок 2.1 – Время шифрования и расшифрования ASCII

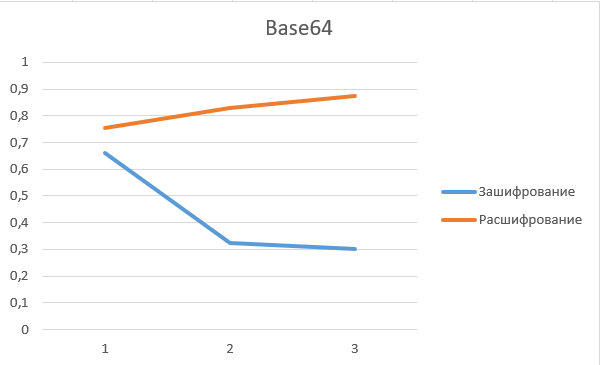


Рисунок 2.2 – Время шифрования и расшифрования Base64

# Вывод

В ходе лабораторной работы были изучены сверхвозрастающие и нормальные последовательности, а также алгоритмы шифрования и расшифрования рюкзаком. Было разработано приложение, реализующее генерацию сверхвозрастающей последовательности, вычисление открытого ключа (нормальной последовательности) из закрытого (сверхвозрастающей последовательности), а также выполняющее шифрование и расшифрование алгоритмом рюкзака.