Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №10**

Исследование ассиметричных шифров RSA и Эль-Гамаля

Выполнил:

Студент 3 курса 5 группы ФИТ

Коршун Никита Игоревич

2024

1. **Зависимости в вычислении параметра у**

В рамках задания лабораторной работы необходимо было выполнить анализ скорости вычисления параметра *y* в выражении представленноом на формуле 1.1.

 (1.1)

Для этого использовался программный код, представленный в листинге 1.1

BigInteger a = new BigInteger(20);

int num = 100000;

BigInteger n;

string N = "618970019642690137449562111875465465468789756132135486794654132123132416489798751321313546618970019642690137449562111875465465468789756132135486794654132123132618970019642690137449562111875465465468789756132135486794654132123132416489798751321313546547897566432132154987655432123456541648979875132131354654789756643213215498765543212345655478975664321321549876554321234565";

BigInteger.TryParse(N, out n);

Stopwatch stopwatch = new Stopwatch();

BigInteger result;

for (int i = 5; i < 35; i += 5)

{

stopwatch.Reset();

stopwatch.Start();

result = BigInteger.Pow(i, num) % n;

stopwatch.Stop();

Console.WriteLine($"y = {result}\t({stopwatch.ElapsedMilliseconds} ms)");

//num += 100000;

}

Листинг 1.1 – Генератор *y*

В первом случае необходимо было проанализировать зависимость от параметра *a*, результат представлен на рисунке 1.1.

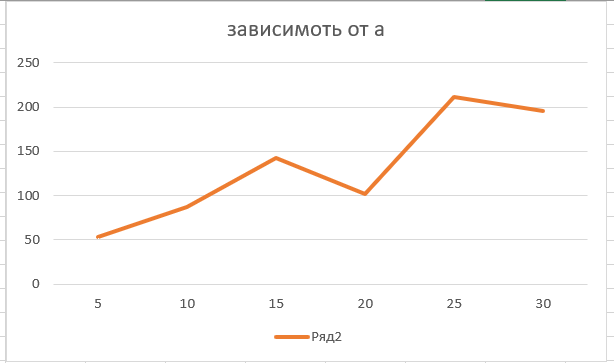


Рисунок 1.1 – Зависимость времени от параметра *a*

Во втором случае зависимость от числа x, результат представлен на рисунке 1.2.

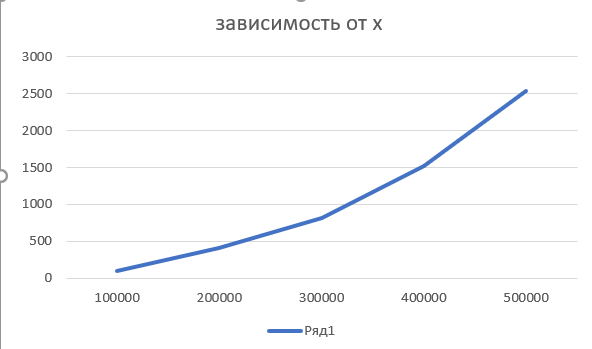


Рисунок 1.2 – Результат работы программы

В третьем случае необходимо было проанализировать влияние при *n* равном 1024 и 2048 битов соответственно отображено на рисунке 1.3.

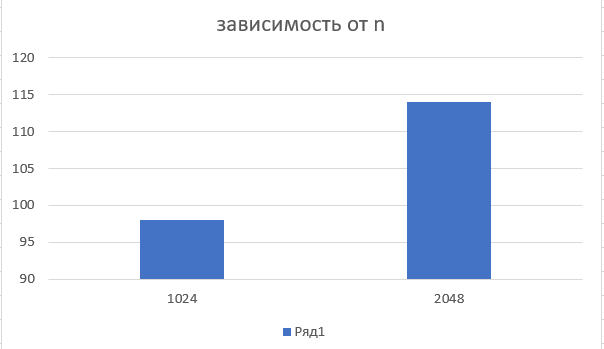


Рисунок 1.3 – Зависимость от n

Как мы можем наблюдать, наиболее влиятельным на время вычисления является именно параметр *n*.

1. **Алгоритм шифрования RSA**

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).

Наконец, расширенный алгоритм Евклида используется для вычисления третьего компонента ключа: ключа расшифрования *d* такого, что выполняется условие представленное в формуле 1.2.

 (1.2)

Таким образом, сформирован ключ, состоящий из трех чисел, которые в свою очередь образуют две вышеупомянутые взаимосвязанные части: открытый (публичный) ключ (*e*, *n*) и тайный ключ (*d*, *n*; на самом деле, как видим, тайным здесь является лишь первое из пары чисел).

Для зашифрования/расшифрования используется ключ получателя: отправитель шифрует сообщение открытым ключом, а получатель расшифровывает шифртекст своим тайным ключом.

Если шифруется сообщение *М*, состоящее из r блоков: *m1*, *m2*, …, *mi*, …, *mr*, то шифртекст *С* будет состоять из такого же числа блоков, представляемых числами из формулы 1.3.

 (1.3)

В коде, для зашифрования используется метод из класса RSACryptoServiceProvider библиотеки RSA, вызов метода представлен в листинге 1.2

public static byte[] EncryptRSA(byte[] plaintext, RSAParameters publicKey)

{

var stopWatch = new Stopwatch();

stopWatch.Start();

var rsa = new RSACryptoServiceProvider();

rsa.ImportParameters(publicKey);

stopWatch.Stop();

Console.WriteLine($"RSA Encrypt:\t\t{stopWatch.ElapsedTicks} ticks ({stopWatch.ElapsedMilliseconds} ms)");

return rsa.Encrypt(plaintext, true);

}

Листинг 1.2 – Зашифрование RSA

Для расшифрования, соответственно, также используется метод данного класса, результат работы программы представлен на рисунке 1.4

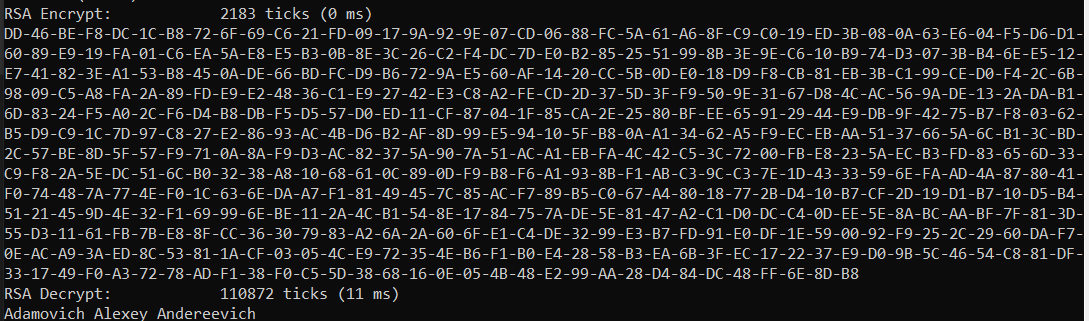


Рисунок 1.4 – Результат работы алгоритма RSA

Можно заметить, насколько шифротекст увеличивает размер передваемого сообщения, что, очевидно, не очень оптимально. На рисунке 1.5 показан график зависимости времени шифрования в зависимости от длины сообщения

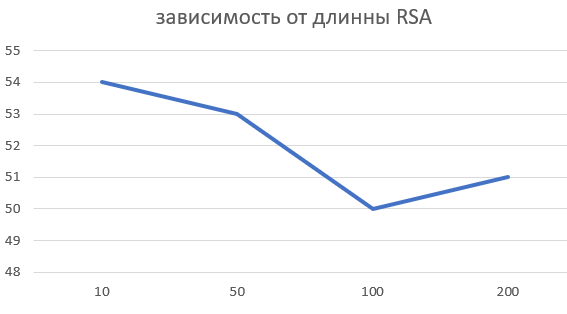


Рисунок 1.5 – График зависимости RSA

Как мы видим, в RSA зависимость времени шифрования от длинны сообщения фактически отсутствует и ее изменение можно счесть за погрешность.

1. **Алгоритм шифрования Эль-Гамаля**

Он может быть использован для решения трех основных криптографических задач: для зашифрования/расшифрования данных, для формирования цифровой подписи и для согласования общего ключа. Кроме того, возможны модификации алгоритма для схем проверки пароля, доказательства идентичности сообщения и другие варианты.

Как подчеркивалось выше, безопасность алгоритма Эль-Гамаля, как и безопасность алгоритма Диффи – Хеллмана, основана на трудности вычисления дискретных логарифмов. Алгоритм Эль-Гамаля фактически использует схему Диффи – Хеллмана, чтобы сформировать общий секретный ключ для абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ. Формула (1.4) для генерации ключевой информации представлена

 (1.4)

Код для зашифрования сообщения алгоритмом Эль-Гамаля представлен на листинге 1.2.

var stopWatch = new Stopwatch();

stopWatch.Start();

BigInteger y = 0, a = 0;

BigInteger[] b = { };

string encData;

char[] array;

encData = "";

y = BigInteger.ModPow(g, x, p);

array = plaintext.ToCharArray();

int k = 20;

a = BigInteger.ModPow(g, k, p);

b = new BigInteger[array.Length];

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

{

b[i] = BigInteger.Remainder(BigInteger.Multiply(BigInteger.Pow(y, k), array[i]), p);

encData = encData + b[i].ToString();

}

Console.WriteLine("\nEncrypted data: " + encData);

stopWatch.Stop();

Console.WriteLine($"El-Gamal Encrypt:\t{stopWatch.ElapsedTicks} ticks ({stopWatch.ElapsedMilliseconds} ms)");

return b;

Листинг 1.2 – Зашифрование Эль-Гамаля

В результате работы данного алгоритма вывод программы представлен на рисунке 1.6.

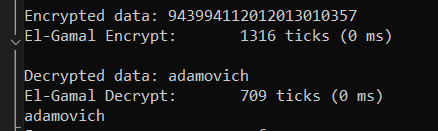


Рисунок 1.6 – Вывод Эль-Гамаль

Зависимость времени расшифрования в зависимости от длины сообщения представлен на рисунке 1.7.

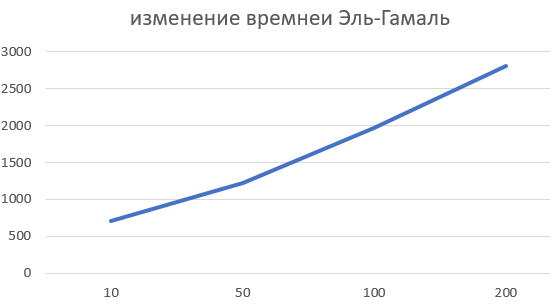


Рисунок 1.7 – Зависимость времени расшифрования Эль-Гамаля

В отличии от RSA в данном алгоритме прямая зависимость времени работы алгоритма от длины сообщения. Что показывает не самую высокую эффективность на больших размерах сообщений, но если длина сообщений имеет очень большой разброс, то это более эффективно чем RSA.

Однако, стоит отметить, что длинна крипто текста в Эль-Гамале превышает длину открытого текста как минимум в 1.5 раза

**Вывод:** Шифры на основе ассиметричных ключей являются актуальными до сих пор и активно используются для генерации электронных подписей, заверения сертификатов, благодаря сложности подбора значений благодаря огромной ресурсоемкости вычислений логарифмирования.